ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 3, May2012

Comparative Study of Software Module Clustering Algorithms: Hill-Climbing, MCA and ECA

Kishore C **Srinivasulu Asadi** **Anusha G**
Department of Information Techonolgy, Sree Vidyanikethan Engineering College, Tirupathi, India
kishore.btech16@gmail.com
Department of Information Techonolgy, Sree Vidyanikethan Engineering College, Tirupathi, India
Srinu_asadi@yahoo.com
Department of Information Techonolgy, Sree Vidyanikethan Engineering College, Tirupathi, India
anua13@gmail.com

***Abstract*—Most interesting software systems are large and complex, and as a consequence, understanding their structure is difficult. One of the reasons for this complexity is that source code contains many entities (e.g., classes, modules) that depend on each other in intricate ways (e.g., procedure calls, variable references). Additionally, once a software engineer understands a system's structure, it is difficult to preserve this understanding, because the structure tends to change during maintenance. Research into the software clustering problem has proposed several approaches to deal with the above issue by defining techniques that partition the structure of a software system into subsystems (clusters). Subsystems are collections of source code resources that exhibit similar features, properties or behaviors. Because there are far fewer subsystems than modules, studying the subsystem structure is easier than trying to understand the system by analyzing the source code manually. The contribution of the work included: In previous single objective search problem has been developed. Our ultimate goal is to develop the multi-objective search problem and compared with Single-objective search problem. The results of this empirical study provide strong evidence to support the claim that the multi-objective approach provides significantly better solutions than the existing single-objective approach.**

***Index Terms*-SBSE, Module Clustering, multi-objective optimization, evolutionary computation, MCA, ECA.**

I. INTRODUCTION

Software module clustering is an important and challenging task in software engineering. We believed that a well-modularized software system is easy to develop and maintain [2], [5], [7]. A well module software structure is regarded as one that has a high cohesion and low coupling. Here we are focused on automated techniques for suggesting software clustering and also for delimiting boundaries between modules that maximize the degree of cohesion and minimize the degree of coupling.

There are different ways to approach the software module clustering problem. Following Mancoridis et al., who first suggested the search-based approach to module clustering. Here we follow 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" guides a search-based optimization algorithm.

In previous work on software module clustering has used a single-objective formulation of the problem [1], [4], [9], [8], [6], [3], [10]. Single objective is taken as, integrating the twin objectives of high cohesion and low coupling i.e., Modularization Quality (MQ). They used a search based optimization algorithm for single objective approach is hill-climbing algorithm [3]. There is a natural tension between the objective of achieving low coupling and the objective of achieving high cohesion when defining module boundaries. These two aspects of the system will often be in conflict. Therefore, any attempt to conflate cohesion and coupling into a single objective may yield suboptimal results.

This paper introduces the first Pareto optimal multi-objective formulation of automated software module clustering, presenting results that show how this approach can yield superior results to those obtained by the single-objective formulation.

The primary contributions of the paper are as follows:

1. The multi-objective paradigm for automated software module clustering is introduced. Two formulations of the multiple objective approach are studied: the Equal-size Cluster Approach (ECA) and the Maximizing Cluster Approach (MCA).
2. An empirical study into the effectiveness and performance of the single and multi-objective formulations of the problem is presented.

The rest of this paper is organized as follows: Section 2 describes how the software modules clustering process will be done. Section 3 describes single-objective search i.e., hill-climbing algorithm. Section 4 introduces multi-objective paradigm of module clustering. Section 5 presents the findings of the empirical study, while section 6 concludes.

II. SOFTWARE MODULE CLUSTERING

Software clustering problem has defined techniques that partition the structure of a software system into subsystems. Subsystems are collection of source code resources that exhibit similar feature, properties or behavior. Because there are far fewer subsystems than modules, studying the subsystem structure is easier than trying to understand the system by analyzing the source code manually.

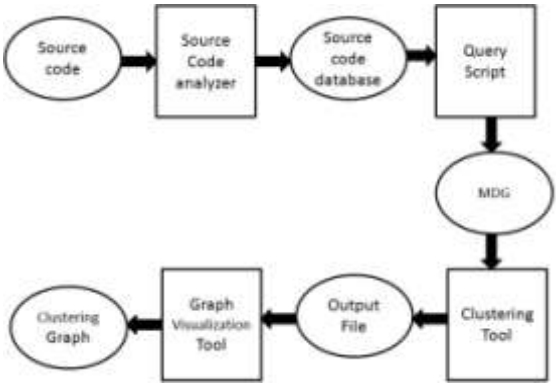

Figure 1. Software Module Clustering Process

Figure 1 shows the process that how the software module clustering process will be done.

The first step in the clustering process is to extract module-level dependencies from the source code and store the resultant information in a database by using source code analysis tool. After all the module level dependencies have been stored in a database, a script is executed to query the database, filter the query, 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 represent the module (classes) in the system as nodes, and the relations between modules as weighted directed graph.

Once the MDG is created, Bunch 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 partitioned MDG is created, we use graph drawing tools such as Graphviz (dotty) to visualize the results.

III. SINGLE-OBJECTIVE SEARCH APPROACH

Hill-Climbing clustering algorithm [3] is the single-objective search algorithm. It starts with a random partition of the MDG.

Hill-Climbing(MDG M; Integer popSz; Threshold t)
Let P be a set of random partitions of M containing popSz members
Let B be the best partition of M, initialized to 0
Let $max_{mq} = -\infty$
Foreach partition pϵ P do
    Let currentP = p
    Let nextP = ClimbHill(currentP, t)
    While MQ(nextP)>MQ(currentP) do
        CurrentP←nextP
        nextP = ClibmHill(currentP, t)
    end
    Let mq_{hc} = MQ(currentP)
    If $mq_{hc} > max_{mq}$ then
        B←currentP
        $max_{mq} \leftarrow mq_{hc}$
    end
end
return (partition B)

ClimbHill(Partition P; threshold n)
Let BestP = P
Let neighbourEvalCnt = (P.numClusters) x (P.MDG.numNodes) x n
Let N be the set of neighbors of P
N← randomize(N)
Let max_{mq} = MQ(P)
Let count = 0
Let improved = false
Foreach neighbor nϵ N do
    Count ← count+1
    Let partition C = P.applyNeighbor(n)
    Let mq_n = MQ(C)
    If $mq_n > max_{mq}$ then
        BestP ← C
        $max_{mq} \leftarrow mq_n$
        improved = true
    end
    if(count ≥ neighborEvalCnt and improve = true then
        return (partition BestP)
    end
end
return (partition BestP)

Figure 2. Hill-Climbing Algorithm

Modules from this partition are then systematically rearranged in an attempt to find an "improved" partition with a higher MQ. If a better

partition is found, the process iterates, using the improved partition as the basis for finding even better partitions. This hill-climbing approach eventually converges when no additional partitions can be found with a higher MQ.

Hill-Climbing algorithms move modules between the clusters of a partition in an attempt to improve MQ. This task is accomplished by generating a set of neighboring partitions (NP). A partition NP and P are neighbor with each other if N only if NP and P are same except the single cluster of a partition P is in a different cluster in partition NP. If partition P contains m nodes and k clusters, the total number of neighbors is O(n. k). It should be noted that for many partitions of an MDG the number of neighbors is exactly n.k. However, if a partition contains clusters with 1 or 2 nodes, the total number of distinct neighbors is slightly less.

The Hill-Climbing(...) function manages the individual partitions in the population, and the ClimbHill(...) function takes a particular partition and "improves" it using the neighboring technique described above.

There is a limitation in Hill-Climbing clustering algorithm i.e., it is not practical to use with the systems that have more than 15 modules.

IV. MULTI-OBJECTIVE SEARCH APPROACH

Here we proposed the first pareto optimal multi-objective formulation of automated software module clustering, presenting results that show how this approach can yield superior results to those obtained by the single objective formulation. Each set of objectives leads to a different multi-objective formulation of the problem. In this paper, two sets of objectives will be considered: The Maximizing Cluster Approach and the Equal-size Cluster Approach. These are explained below:

A. The Maximizing Cluster Approach

The aim of Maximizing Cluster Approach is to capture the attributes of a good clustering. It will have maximum possible cohesion and minimal possible coupling. But it should not put all the modules into a single cluster and not produce a series of isolated clusters.

The objectives of MCA are as follows:

- The sum of intra-edges of all clusters (Maximizing)
- The sum of inter-edges of all clusters (Minimizing)
- The number of clusters (Maximizing)
- MQ (Maximizing)
- The number of isolated clusters (Minimizing)

The inter-edges, inter-edges, MQ are used to measure the quality of the system partitioned. An isolated cluster is a cluster which contains only one module. MQ is a well-studied objective function, so MQ is included as objective of MCA. 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 as an objective of MCA.

B. The Equal-Size Cluster Approach

The ECA produce a modularization that contains clusters having roughly equal size. This approach decomposes the software system into roughly equal-size modules. This tends to mitigate against small isolated clusters and also avoid the presence of one large "god-class" like structure.

The objectives of ECA are as follows:

- The sum of intra-edges of all clusters (Maximizing)
- The sum of inter-edges of all clusters (Minimizing)
- The number of clusters (Maximizing)
- MQ (Maximizing)
- The difference between the maximum and minimum number of modules in a cluster (Minimizing)

Pareto optimal multi-objective formulations of automated software module clustering have been implemented by using genetic algorithm. A generic genetic algorithm is presented in figure 3.

```
Set generation number, m=0
Choose the initial population of candidate solutions, P(0)
Evaluate the fitness for each individuals of P(0), F(P_i(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;
```

Figure 3: A generic genetic algorithm

V. EMPIRICAL STUDIES

This section explains about comparison among the three algorithms. i.e., single-objective formulation known as Hill-Climbing described in section III and multi-objective formulations of the clustering problem, known as MCA and ECA described in section IV. In order to evaluate the effectiveness of single-objective and multi-objective approaches, a set of experiments was performed on 17 real-world module clustering problems which are tabulated in table 1. By taking the following criteria's we perform comparison.

A. MQ Value as assessment criterion:

How well does the two-archive multi-objective search perform when compared against the Bunch approach using the MQ value as the assessment criterion? Table 2 presents the comparison among the Hill-Climbing, MCA and ECA approaches by taking MQ value as assessment criterion. There is a good evidence to suggest that for unweighted problems, the hill-climbing algorithm outperformed the MCA approach. That is, the hill-climbing algorithm gives higher values for MQ in six from seven problems. For weighted MDG problem, MCA approach outperforms the hill-climbing approach. That is, MCA beats the hill-climbing algorithm in 7

Table 1. The Systems Studied

	Name	Nodes	Edges	Description
unweighted	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
weighted	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

from 10 problems. There is no evidence to suggest that ECA approach is outperformed by the hill-climbing approach for unweighted graphs. There is strong evidence to suggest that the ECA approach outperforms the hill-climbing for weighted MDGs. In all cases, The ECA approach outperforms the MCA approach. Overall study says that Hill-climbing approach performs superior values for unweighted graphs than the MCA approach. For weighted graphs, multi-objective approach (particularly ECA approach) can produce better values than hill-climbing approach.

B. Cohesion and Coupling as assessment criterion:

How well do the two-archive algorithm and the Bunch perform at optimizing each of the two primary software engineering objectives of low coupling and high cohesion? Table 3 presents the comparison among the Hill-Climbing, MCA and ECA approaches by taking MQ value as assessment criterion. The comparison of MCA and Hill climbing is somewhat inconclusive for unweighted graphs. For weighted MDGs, the MCA approach outperforms the Hill-climbing approach in all cases with statistical significance. The results provide strong evidence to suggest that the ECA approach outperforms the hill-climbing approach for both weighted and unweighted graphs. In all cases ECA approach is preferable to the MCA approach. Overall study says that multi-objective approach outperforms the hill-climbing approach in producing solution clusterings with both higher cohesion and lower coupling for weighted graphs.

C. Pareto optimality as assessment criterion:

How good is the Pareto front achieved by the two approaches? Multi-objective formulations can be expected to outperform the single single-objective formulation since they are designed to produce good approximations to the Pareto front, whereas the single-objective approach is not.

Table 4, 5 and 6 displays the dominance relationship for the results obtained from all three approaches. This dominance relationship is used to compare any two solutions in multi-objective space. The three software engineering objectives considered are the intra-edge and the inter-edge measurement, and, for backward compatibility with work on single-objective formulations, the MQ value obtained. In these tables, A denotes the hill-climbing algorithm, B denotes the MCA, and C denotes the ECA. The heading NXY denotes the number of solutions generated by X that are dominated by solution in Y. In comparison, X is better than Y if NXY is small and NYX is large.

Table 4 shows that the number of solutions produced by hill climbing outperforms MCA for unweighted problems (six from seven problems), while, in weighted systems, the MCA outperforms the hill-climbing algorithm in all problems. Table 5 provides strong evidence that ECA outperforms hill climbing for both weighted and unweighted MDGs. Table 6 shows that ECA comfortably outperforms MCA in all of the problems studied.

Table 2. Comparison of algorithms by taking MQ as assessment criterion.

	Name	MCA	Hill-Climbing	ECA
unweighted	mutunis	2.294	2.249	2.314
	ispell	2.269	2.337	2.339
	rcs	2.145	2.218	2.239
	bison	2.416	2.639	2.648
	grappa	11.586	12.676	12.578
	bunch	12.145	13.536	13.455
	incl	11.811	13.568	13.511
weighted	icecast	2.401	1.779	2.654
	gnupg	6.259	4.869	6.905
	inn	7.421	6.720	7.876
	bitchx	3.572	2.465	4.267
	xntp	6.482	6.655	8.168
	exim	5.316	5.199	6.361
	mod_ssl	8.832	7.906	9.749
	ncurses	10.211	9.836	11.297
	lynx	3.447	3.488	4.694
	nmh	6.671	7.012	8.592

D. Computational Effort:

This criterion compares the effort required to solve the clustering problem using the traditional hill-climbing approach and the new multi-objective approaches introduced in this paper. The results indicate that there is a trade-off between effort and quality of results. That is, the number of evaluations required to achieve the better results of the multi-objective approach is two orders of magnitude greater than that required for the hill climber. However, even if we allow the Hill Climber the same number of fitness evaluations as the multi-objective approaches, the results for the multi-objective approaches are still typically better than those obtained by the Hill Climber

Table 3. Comparison of algorithms by taking Cohesion and Coupling as assessment criterion

	Name	MCA		Hill-Climbing		ECA	
		Intra-edges	Inter-edges	Intra-edges	Inter-edges	Intra-edges	Inter-edges
unweighted	mutunis	24.633	-64.733	22.600	-68.800	27.000	-60.000
	ispell	23.100	-159.800	25.833	-154.333	30.033	-145.933
	rcs	45.133	-235.733	35.033	-255.933	47.567	-230.867
	bison	40.367	-277.267	40.600	-276.800	52.800	-252.400
	grappa	84.767	-420.467	81.900	-426.200	101.167	-387.667
	bunch	73.567	-580.867	100.867	-526.267	111.700	-504.600
	incl	91.767	-536.467	140.967	-438.067	140.200	-439.600
weighted	icecast	1609.900	-7636.200	733.467	-9389.070	1643.167	-7569.670
	gnupg	1104.733	-5192.530	887.200	-5627.600	1494.167	-4413.670
	inn	771.633	-6176.730	554.233	-6611.530	1336.900	-5046.200
	bitchx	7644.633	-35938.700	3166.267	-44895.500	7840.600	-35546.800
	xntp	733.800	-4460.400	447.033	-5033.930	1117.967	-3692.070
	exim	3279.300	-12347.400	1004.267	-16897.500	3146.567	-12612.900
	mod_ssl	2911.733	-12138.500	1101.633	-15758.700	3476.800	-11008.400
	ncurses	574.433	-3071.130	368.700	-3482.600	806.367	-2607.270
	lynx	2428.567	-23150.900	1567.800	-24872.400	3730.633	-20546.700
	nmh	2032.267	-19921.500	1120.233	-21745.500	2704.600	-18576.800

.

Table 4. Results of Dominated Comparison

	Name	N_{AB}	N_{BA}
unweighted	mutunis	23	19
	ispell	7	30
	rcs	2	21
	bison	0	29
	grappa	0	23
	bunch	0	30
	incl	0	30
weighted	icecast	30	0
	gnupg	30	0
	inn	30	0
	bitchx	30	0
	xntp	23	10
	exim	30	0
	mod_ssl	30	0
	ncurses	30	0
	lynx	26	12
	nmh	13	0

Table 5. Results of Dominated Comparison

	Name	N_{AC}	N_{CA}
unweighted	mutunis	23	0
	ispell	24	13
	rcs	30	3
	bison	28	17
	grappa	27	0
	bunch	16	11
	incl	22	27
weighted	icecast	30	0
	gnupg	30	0
	inn	30	0
	bitchx	30	0
	xntp	30	0
	exim	30	0
	mod_ssl	30	0
	ncurses	30	0
	lynx	30	0
	nmh	30	0

Table 6. Results of Dominated Comparison

	Name	N_{BC}	N_{CB}
unweighted	mutunis	19	0
	ispell	30	0
	rcs	27	0
	bison	30	0
	grappa	30	0
	bunch	30	0
	incl	30	0
weighted	icecast	29	0
	gnupg	30	0
	inn	30	0
	bitchx	24	0
	xntp	30	0
	exim	30	0
	mod_ssl	30	0
	ncurses	30	0
	lynx	30	0
	nmh	29	0

VI. Conclusion And Future Work

This paper introduces multi-objective formulations i.e., ECA and MCA. Then the results obtained by applied on 17 real-world module clustering problems are compared with single-objective formulation i., Hill-Climbing algorithm. The results of empirical study show that multi-objective formulation gives better solution than the single-objective formulation. Especially Equal-Size Cluster Approach is able to produce better solutions than the existing single-objective solution.

In future work we could consider more objectives like foot print size and communication band width for better modularization.

References

[1] S. Mancoridis, B.S. Mitchell, C. Rorres, Y.-F. Chen, and E.R. Gansner, "Using Automatic Clustering to Produce High-Level System Organizations of Source Code," Proc. Int'l Workshop Program Comprehension, pp. 45-53, 1998.

[2] L.L. Constantine and E. Yourdon, Structured Design. Prentice Hall, 1979.

[3] 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.

[4] 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.

[5] R. Pressman, Software Engineering: A Practitioner's Approach, third ed. (European adaptation (1994), adapted by Darrel Ince). McGraw-Hill, 1992.

[6] B.S. Mitchell and S. Mancoridis, "Using Heuristic Search Techniques to Extract Design Abstractions from Source Code," Proc. Genetic and Evolutionary Computation Conf., pp. 1375-1382, July 2002.

[7] I. Sommerville, Software Engineering, sixth ed. Addison-Wesley, 2001.

[8] 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.

[9] 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.

[10] M. Harman, S. Swift, and K. Mahdavi, "An Empirical Study of the Robustness of Two Module Clustering Fitness Functions," Proc. Genetic and Evolutionary Computation Conf., pp. 1029-1036, June 2005.

Authors Biography

Kishore C received the B.Tech Information Technology from JNTUA, Anantapur, India in 2010 and pursuing his Master's degree in Software Engineering from the JNTUA, Anantapur, India. His research areas are Software Engineering, Data warehousing and Data Mining, Database Management Systems and Cloud Computing. He has published 5 papers in International Journals and Conferences. Some of his publications appear in IJCSIT and IJARCSSE digital libraries.

Asadi Srinivasulu received the B Tech Computer Science Engineering from Sri Venkateswara University, Tirupati, India in 2000 and M.Tech with Intelligent Systems in IT from Indian Institute of Information Technology, Allahabad (IIIT) in 2004 and he is pursuing Ph.D in CSE from J.N.T.U.A, Anantapur, India. He has got 10 years of teaching and industrial experience. He served as the Head, Dept of Information Technology, S V College of Engineering, Karakambadi, Tirupati, India during 2007-2009. His areas of interests include Data Mining and Data warehousing, Intelligent Systems, Image Processing, Pattern Recognition, Machine Vision Processing and Cloud Computing. He is a member of IAENG, IACSIT. He has published more than 35 papers in International Journals and Conferences. Some of his publications appear in IEEE Xplore, IJCA, IJCSET, IJCSIT and IJARCSSE digital libraries. He visited Malaysia and Singapore.

Anusha G received the B.Tech Information Technology from JNTUA, Anantapur, India in 2010 and pursuing his Master's degree in Software Engineering from the JNTUA, Anantapur, India. Her research areas are Software Engineering, Data warehousing and Data Mining, Database Management Systems and Cloud Computing. she has published 3 papers in International Journals and Conferences.