An Efficient Scheme for Candidate Solutions of Search-Based Multi-objective Software Remodularization

Amarjeet Prajapati(✉) and Jitender Kumar Chhabra

Computer Engineering Department, NIT Kurukshetra, Kurukshetra, India
amarjeetnitkkr@gmail.com, jitenderchhabra@gmail.com

Abstract. Multi-objective search-based software remodularization approaches are used to rearrange the software elements into modules by optimizing several quality criteria. These search-based approaches can find out better quality regrouping solutions compared to the traditional (analytical based) remodularization, if the suitable encoding for the candidate solutions is used. In this paper, we propose an efficient encoding scheme for candidate solutions and use this scheme to remodularize the object-oriented software using genetic based multi-objective evolutionary algorithm. This proposed representation helps in improving human-computer interaction, and semantic based, efficiently-designed and error-free information gets transferred to the computing system through it. To assess the effectiveness of the proposed approach, we evaluate it over six real-world software systems of different characteristics. Further, the approach is compared with the existing encoding scheme (i.e., GNE representation scheme). Experiments show that the proposed approach produces better results in terms of quality, convergence speed and execution time compared to GNE representation scheme.

Keywords: Software remodularization · Search based software engineering · Multi-objective optimization · Object-oriented systems

1 Introduction

Software Remodularization: Software systems, in particular, object-oriented systems are initially designed and created in a modular way. However, over time, modularity of a system often degrades due to improper placement of software elements into the modules [1]. The ill-modular structure of software system makes it difficult to understand, maintain and evolve [2]. To improve the modularity of software system the maintainers reorganize the software components into the modules and such process is generally known as software remodularization [3]. Software remodularization problem generally involves large set of conflicting criterion, competing constraints, ambiguous and vague information [4]. Hence, solving remodularization problem is very complex and time consuming task. Consider the problem of finding a remodularization of software system that satisfies a given modularity criterion. In such a problem, there can be very large/infinite remodularization alternatives and different sets of modularization can be good solutions. Furthermore, modularization needs to satisfy various competing

S. Yamamoto (Ed.): HIMI 2016, Part I, LNCS 9734, pp. 296–307, 2016.
DOI: 10.1007/978-3-319-40349-6_28

constraints related to the modularity. In such case, identification of an overall good modularization solution is highly desirable for the maintainer, but not easy to obtain through an automated process.

Software remodularization techniques can be broadly categorized into analytical-based and search-based remodularization. Several analytical analysis based techniques have been presented in the research literature for software remodularization [3, 5–7]. These approaches guarantee to find an optimal solution, but require exponential computational time. Hence, such approaches can be more suitable for small size problems. On the other hand the search-based remodularization approaches do not guarantee the best optimal solution, but are able to find near optimal solution in a reasonable time.

This paper uses the search-based remodularization approach and primarily concerns with the problem of rearranging the source code classes into a set of modules of a system whose modularization has degraded due to the maintenance. For such a system some classes may no longer be in suitable modules and thus remodularization of the system become highly useful for improving the software quality. Our aim is therefore to search the whole space of possible modularizations to see if there exists a better grouping of classes in various modules.

Search-Based Multi-objective Software Remodularization: The term "Search-Based Software Engineering" was first given by Harman and Jones [8]. Since then lot of research has been carried out in this direction. As addressed by Harman and Jones [8], the problems to be solved by the SBSE usually require more efforts, the solution is highly intricate, and the software developer is ready to wait for the output. The search-based approach provides a lower human cost solution, freeing the software developer to work on other issues that require creativity and imagination.

To solve any problem using search-based technique, problem needs to be reformulated as a search-based optimization problem. For that, it will be necessary to define the following there key ingredients (1) representation of candidate solution; (2) fitness function; and (3) manipulator operators. Proper design of these ingredients has major influence on the performance of search-based software remodularization. The main contribution of this paper is a new and efficient representation of candidate solutions, which helps in improving the human computer interaction (HCI) as software developers (humans) interact with the computer based optimization process through this representation. An efficient representation eases the process of inputting the suitable data to the computing systems ensures the transfer of correct data from humans to computers and has a major impact on performance of computing of this data at the machine end [9].

Good representation of the candidate solutions is critical to the convergence speed of the search-based technique and the quality of obtained results [10]. Most of the existing search-based remodularization approaches use the integer based representation namely GNE (Group Number Encoding) for candidate solution [4, 10–14]. It is most widely used solution representation approach in both single and multi-objective search-based software remodularization. It is represented as an n-sized integer vector, where the value $0 < c_i \leq n$ of the ith class indicates the cluster which the ith class is assigned. A remodularization solution with the same value for all the classes means that all classes are placed in the same cluster, while a solution with all different values (from

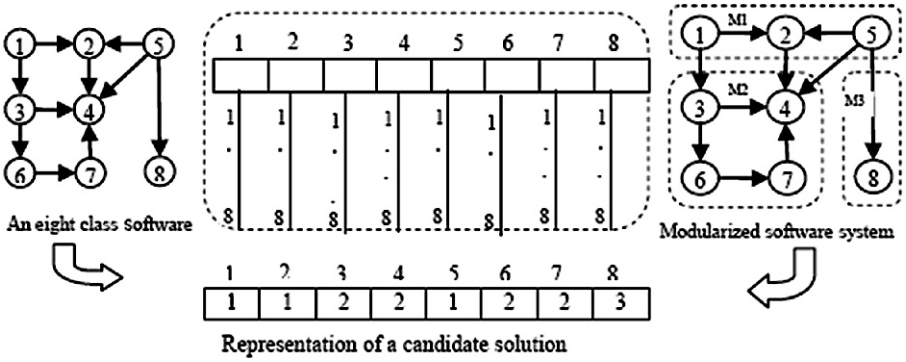

Fig. 1. Illustration of candidate solution using GNE representation

1 to n) means that each classes is in a separate module. The disadvantage of this representation is that it is highly redundant. Figure 1 illustrates an example of the GNE representation.

Above-mentioned integer-based representation approach has been successfully used in many research papers for solving the search-based software remodularization problems. However, major drawback of this representation is that it generates many redundant solutions, which increases the search space and hence execution time. For example, consider software consisting of 7 classes being grouped into 3 modules, the vector (1,1,2,1,2,3,3), (3,3,1,3,1,2,2) and (2,2,3,2,3,2,2) all represent same modularization solution, which places classes 1,2 and 4 in one module, classes 3 and 5 in another module and which places classes 6 and 7 in a third module. Hence, such similar modularization solutions increase the search space.

To minimize the redundancy, we propose a new representation technique in search-based multi-objective software remodularization framework. The new representation is an important task which facilitates an efficient human and computer interaction. Transformation of the software structure into new transformation is a manual or semi automated process usually and optimization is a computer based processing task. The proposed representation requires a clear comprehension of semantics of the structure so that a suitable human computer interaction (HCI) can be planned. The proposed approach for software remodularization is able to produce better solution with faster convergence compared to the previous approach.

The rest of the paper is organized as follows. Section 2 presents relevant research literature. Section 3 describes a framework of the proposed search-based multi-objective software remodularization. Section 4 presents the experiments, results and analysis. Section 5 concludes with future work.

2 Related Works

A large number of research works have been proposed in the literature to support the automatic software remodularization [3, 4, 7, 14–19]. Most of the existing software remodularization approaches are based on analytical techniques [3, 7, 15, 17] or

search-based optimization techniques [4, 14, 16, 19]. In search-based approaches, once a software system is framed as a search problem, there are many search algorithms can be applied to solve the problem.

The encoding of the candidate solution is an important activity for the performance of the search problem. In the research literature of search-based software remodularization, many representations have been used such as binary code, floating point number, grey code and integer numbers. However, for search-based software remodularization problem, integer number representation has been found to be more suitable than other representations [4]. Here, we discuss those prevailing approaches which are close to our proposed work.

Mancoridis et al. [11] were the first who formulated the software remodularization problem as search based optimization problem and applied Hill-Climbing and Genetic Algorithms to optimize it. They used an integer based representation techniques to represent the candidate solution. Thereafter, Mitchell and Mancoridis [16] used the same representation technique in Hill-Climbing and Genetic Algorithm for development of Bunch, a tool supporting automatic software remodularization. Praditwong et al. [4] also used the same representation in a new evolutionary algorithm named as two-archive multi-objective evolutionary algorithms to address the software remodularization problems. Abdeen et al. [20] also applied evolutionary algorithms with same representation to modularize the source code classes into packages by automatically minimizing the package dependencies of object-oriented software system. Later, Abdeen et al. [13] extended the same work by formulating the problem as a multi-objective optimization and performed optimization using a multi-objective evolutionary algorithm.

Apart from the above search-based software remodularization, there are some more representations in evolutionary clustering literature. In 1998 Falkenauer [21, 22] proposed a encoding for a variable-length genetic algorithm. The encoding is carried out by separating each individual in the algorithm into two parts: c = [l | g], the first part is the element section, whereas the second part is called the group section of the individual. In [23], research each vector is a sequence of real numbers representing the K cluster centers. For an N -dimensional search space, the length of a clustering solution is N*K words, where the first N positions, represent N dimensions of the first cluster centre and next N positions represent those of the second cluster centre, and so on.

3 Proposed Approach

To improve the performance of search-based software remodularization especially multi-objective search-based software remodularization, this paper proposes a new efficient representation technique for the candidate solution. The considered multi-objective search technique is based on the Non-dominated Sorting concepts [24]. The general structure of the proposed search-based multi-objective software remodularization is given in Fig. 2.

The approach is divided into two main parts. In first part, the software system which is to be re-modularized, is parsed and classes, packages and dependencies between all pairs of classes is extracted. Using the extracted information, initial

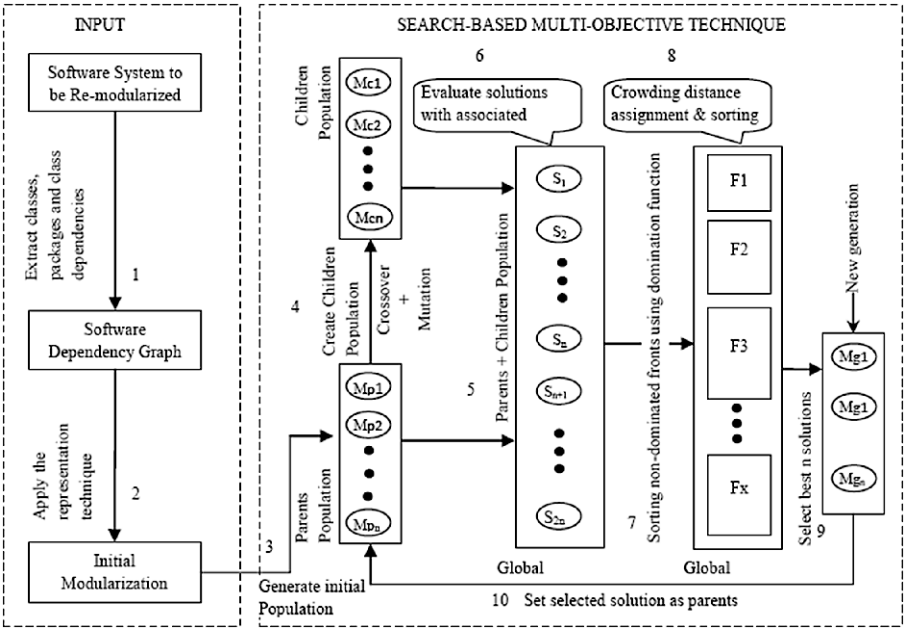

Fig. 2. Overview of search-based multi-objective remodularization process

modularization solution is encoded with the proposed representation technique (in Sect. 3.1). In the second part, the multi-objective evolutionary algorithm (i.e., non-dominated sorting based genetic algorithm) is applied to the initial remodularization solution. The algorithm starts by generating initial population from the initial modularization solution. The child population is generated from the initial population by applying the crossover and mutation operators. Next, parent and children populations are combined and a global population is generated. The global population is evaluated with the associated quality measurement. Now the candidate solutions in the global population are categorized according to their dominance. Non-dominated solutions are given a rank of 1; the candidate solutions dominated only by non-dominated candidate solutions are given a rank of 2; candidate solutions dominated only by the previous are given a rank of 3, and so on. After ranking, the new parent population for further generation is generated from the non-dominated solution. The algorithm evolves a population from generation to generations by applying crossover, mutation, and selection on the candidate solutions.

3.1 Representation of Candidate Solution

For software engineering domain's search-based optimization algorithms, first and critical issue is the representation of candidate solutions [9]. In case of software remodularization problem there is a need to identify each possible correct combination of re-modularizing a software system. Representation must be chosen carefully so that there is one and exactly one candidate representation per modularization.

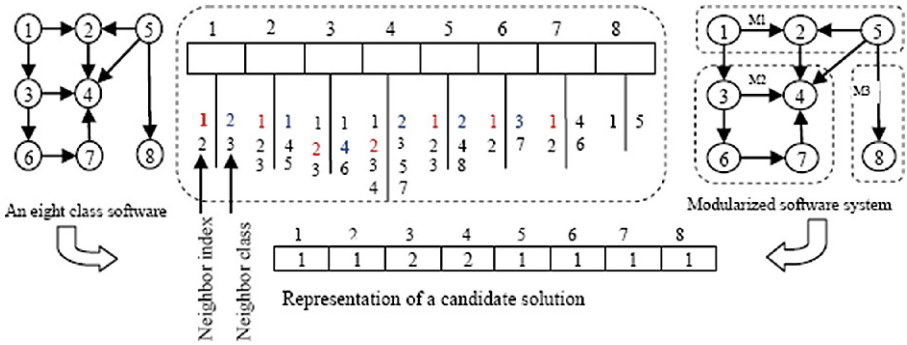

Fig. 3. Illustration of candidate solution representation to an 8 class software remodularization problem.

The approach we adopted in this paper is inspired by the work presented in [25]. In our search-based software remodularization, each modularization solution is represented as a vector of integers (c = [c1, c2, …, cn]), where n is the number of classes in the software dependency graph. In this representation, each ci is an index between 1 and number of topological neighbors for classes i. When ci = 1, class i is located in the same module as its first neighbor; when ci = 2, it is located in the same module as its second topological neighbor; etc. Figure 3 illustrates the proposed neighbor based encoding scheme.

Conceptually speaking, representing the candidate solutions by means of topological neighbors as discussed above, is usually more computationally efficient than using GNE representation scheme described in Sect. 1. The main reason of improved efficiency is that the search space formed by the proposed approach contains reduced number of redundant solutions compared to the GNE representation. This new representation provides a very good interface for human computer interaction. Software developers (humans) prepare this representation by comprehending the semantics of the structure and at the same time this representation helps in feeding error-free, efficiently-designed and less-redundant data to computing framework. Based on this efficient HCI, relevant, correct & desired information gets transferred to the optimization algorithm which then produces the near optimal solutions in lesser time with better quality.

3.2 Population Initialization

The creation of initial population has a major impact on the efficiency of the search-based evolutionary algorithms. The good initialization of population solution can improve the efficiency of the algorithms. If initial population is generated in such a way that their solutions have a better ability to reach an effective optimal solution, evolutionary algorithm converges quickly [25]. The initial population for the proposed approach is generated using a combination of random initialization and modules generated from K-means algorithm [26].

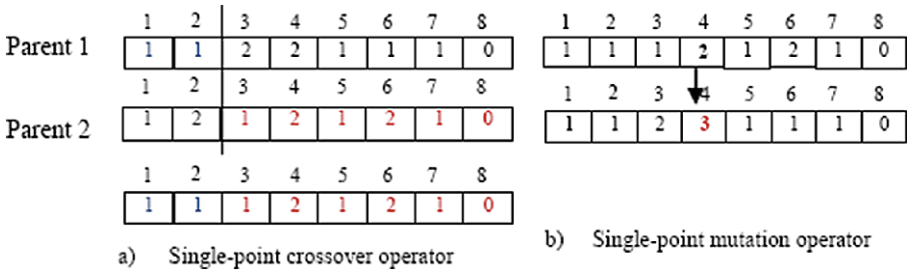

Fig. 4. Illustration of single-point crossover and single-point mutation operators

3.3 Crossover and Mutation

In our search-based software remodularization, for crossover operation, parents are selected using the standard tournament selection method. The crossover process has the following four steps: (1) two parents are selected using the tournament algorithm, (2) a single random point pair is selected at which both parents split, (3) a new individual candidate solution is generated with vector head of first parent and vector tail of second parent, (4) an individual solution in the population is replaced with new individual solution. In mutation operation, a byte is randomly reset to an integer in the feasible set. Figure 4 illustrates the single point crossover and mutation operator.

3.4 Objective Functions

We use five objective functions to optimize our search-based multi-objective: (1) coupling (to minimize), which corresponds to the number of inter-edges (edges between classes in different modules); (2) cohesion (to maximize), which corresponds to the number of intra-edges (edges between classes in the same module); (3) Intra-modular Coupling Density (ICD) [18] (to maximize); (4) number of classes per package (to minimize); (5) number of packages in the system (to minimize).

4 Experiments, Results and Analysis

This section illustrates the results obtained through the experimentation of proposed representation techniques and existing representation techniques by incorporating it into NSGA-II, a multi-objective evolutionary algorithm [24].

4.1 Collecting Results from the Experiments

Since the NSGA-II algorithm is a stochastic optimizer, it can generate different results for the same problem instance from one cycle to another. For this reason, we collect the results for analysis by performing 30 independent simulation runs for each problem instance. As the algorithm is a multi-objective evolutionary optimizer, each running

cycle produces a set of trade-off solutions instead of single solution. However, the purpose of this paper is to demonstrate the usefulness of proposed representation techniques over the software remodularization problem. So we select that single solution which has the highest ICD value in the set of trade-off solutions of each run.

4.2 Studied Software Systems

We performed an empirical study for the proposed coupling schemes over six different object-oriented software systems with different size and characteristics. The main characteristics of the systems examined are given in Table 1. For each system, the version number, number of classes, and number of are mentioned in respective rows. All these software systems are based on the java programming language and are open-source or free-software projects.

Table 1. Characteristics of software systems

Systems	Version	#Classes	#Connections
Jstl	1.0.6	18	20
JavaCC	1.5	154	722
JUnit	3.81	100	276
Java Servlet API	2.3	63	131
XML API DOM	1.0.2	119	209
DOM 4J	1.5.2	195	930

For all software systems, we first analyzed and examined different types of modules and libraries and then removed Omni-present classes such as common libraries, utilities and other domain primitive modules because they do not contribute to any main service.

4.3 Results and Analysis

In this section, we present the software remodularization results obtained by NSGA-II algorithm by incorporating the proposed representation technique. The results are evaluated and compared with existing representation in terms of Intra-modular Coupling Density (ICD), Number of Classes per Packages (NCP) and Execution time. The results are statistically analyzed by two-tailed t-test with 95 % confidence level ($\alpha = 5$ %).

ICD as an assessment criterion: First we present the results of the experiments that compare the ICD values obtained from the proposed approach and the existing approach. Table 2 presents the results comparing proposed and GNE representation scheme. The results clearly indicate that the proposed representation technique performs significantly better to the GNE representation approaches for all software systems. For example, if we consider the Jstl system, ICD value of the proposed representation (i.e., 0.8621) is significantly larger than ICD value of the GNE

representation. Hence, results clearly indicate that the remodularization solutions obtained through the proposed technique have better coupling and cohesion compared to the existing GNE representation.

NCP as an assessment criterion: The results of NCP metric are demonstrated in Table 2. Similar to the ICD metric, results of the proposed approach also perform better in terms of NCP metric. For example, NCP value of the proposed approach on average for all software systems is 4.5 (i.e. 4.5 classes per package), while it is 1.5 for GNE, which is very small. Hence, the proposed approach produces remodularization solutions with better class distribution among the packages compared to GNE representation.

Table 2. Results obtained through proposed and GNE representation

Systems	Intra-modular coupling dependency			Number of classes per package			Execution time (in ms)	
	Proposed	GNE	t-test	Proposed	GNE	t-test	Proposed	GNE
Jstl	0.8621	0.5273	<0.0001	2.5714	1.3254	<0.0001	672	734
Javacc	0.5673	0.3342	<0.0001	4.3561	1.4327	<0.0001	115859	163645
JUnit	0.6341	0.4014	<0.0001	5.8824	1.7857	<0.0001	51956	64338
Java Servlet API	0.5267	0.2756	<0.0001	4.1562	1.5366	<0.0001	8436	9514
XML API DOM	0.5593	0.3668	<0.0001	4.7179	1.8524	<0.0001	149617	183569
DOM 4J	0.5867	0.3237	<0.0001	5.5758	1.4325	<0.0001	171205	231542

Execution time as an assessment criterion: Execution time has a significant influence on the usability of a software remodularization algorithm, especially if the software developers follow an iterative and incremental approach for remodularization the software systems. For example, if the developer uses the slow algorithm to obtain a baseline remodularization solution, modifying the system accordingly may be a time consuming process due to frequent remodularization. Therefore, we evaluated the execution time of the proposed representation technique for software remodularization and compared it with existing GNE. We run algorithm in the environment of Microsoft Windows 7 with 32 bits on Intel Pentium 4 process at 2.4 GHz and 2 GB of RAM and the algorithms are implemented in jMetal 4.5 frameworks. Table 2 shows the running time performance of the remodularization techniques for all eight software systems. The experimental results clearly show that the proposed representation technique for search-based software remodularization is the most time efficient compared GNE representation.

ICD values vs. number of generation: The next experiment is performed to see the growth trend in ICD values with respect to the number of generations for each variant of search-based multi-objective software remodularization (i.e., proposed and GNE representation). Figure 5 demonstrates the ICD growth speed with respect to the number of generations for six problem instances. The vertical axis in these figures shows the ICD value for modularized system. The horizontal axis shows the number of generations. The minimum number of generation is considered 10*N and the maximum

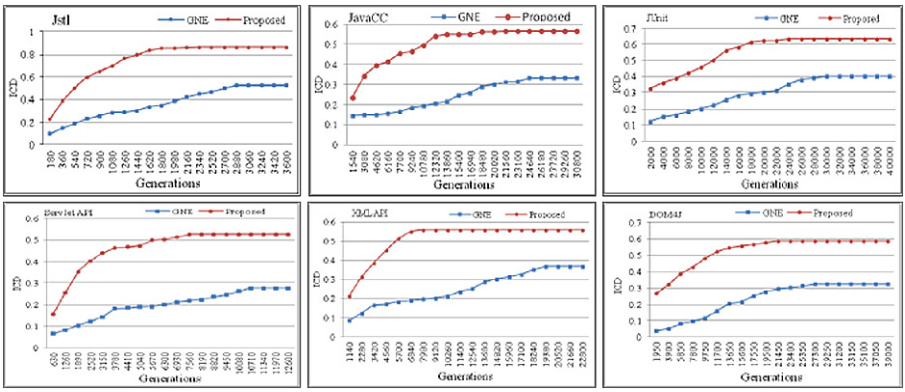

Fig. 5. ICD value vs. number of generation

number of generation is considered 200*N, where N is the total number of classes in the system.

The results demonstrated in Fig. 5 clearly indicate that the growth trend of ICD value of proposed representation scheme for all software system is better compared to the GNE representation scheme. The proposed representation scheme is able to reach the steady state in very small number of generations. However, the GNE representation scheme can only reach at steady state after very large number of generations. For example, after generation 100*N the proposed representation keeps a steady ICD value in all problem instances. However, GNE representation scheme can only reach a steady state until to generation 160*N or later.

5 Conclusion and Future Work

This paper presents a search-based multi-objective optimization approach for software remodularization problem. The approach uses a new and efficient encoding scheme for candidate solutions which ensures a good semantic-based, efficiently-designed, less-redundant and error-free human computer interaction. The approach regroups the classes of the software system by optimizing five objective functions (i.e., coupling, cohesion, Intra-modular Coupling Density (ICD), number of classes per package, and number of packages in the system). The new neighbor based candidate solution representation reduces redundant remodularization solutions from the search space. The approach is evaluated over six real-world software systems and results are compared with the existing integer based representation approach. Experiments show that the proposed approach performs better than the existing integer based representation (i.e., GNE representation) approach in terms of Intra-modular Coupling Density (ICD), number of classes per package, convergence speed as well as execution time.

References

1. Zanetti, M.S., Tessone, C.J., Scholtes, I., Schweitzer, F.: Automated software remodularization based on move refactoring: a complex systems approach. In: Proceedings of 13th International Conference on Modularity (MODULARITY 2014), pp. 73–84. ACM, New York, NY, USA (2014)
2. Bavota, G., Gethers, M., Oliveto, R., Poshyvanyk, D., Lucia, A.D.: Improving software modularization via automated analysis of latent topics and dependencies. ACM Trans. Softw. Eng. Methodol. **23**(1), 1–33 (2014)
3. Anquetil, N., Lethbridge, T.C.: Experiments with clustering as a software remodularization method. In: Proceedings of Sixth Working Conference on Reverse Engineering, pp. 235–255 (1999)
4. Praditwong, K., Harman, M., Yao, X.: Software module clustering as a multi-objective search problem. IEEE Trans. Softw. Eng. **37**(2), 264–282 (2011)
5. Beyer, D., Noack, A.: Clustering software artifacts based on frequent common changes. In: Proceedings of the 13th IEEE International Workshop on Program Comprehension (IWPC 2005), pp. 1092–8138 (2005)
6. Serban, G., Czibula, I.-G.: Restructuring software systems using clustering. In: 22nd International Symposium on Computer and Information Sciences, pp. 1–6 (2007)
7. Maqbool, O., Babri, H.A.: Hierarchical clustering for software architecture recovery. IEEE Trans. Softw. Eng. **33**(11), 759–780 (2007)
8. Harman, M., Jones, B.: Search based software engineering. J. Inf. Softw. Technol. **43**(14), 833–839 (2001)
9. Harman, M. Hierons, R, Proctor, M.: A new representation and crossover operator for search-based optimization of software modularization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1351–1358. Morgan Kaufmann Publishers, New York (2002)
10. Doval, D., Mancoridis, S., Mitchell, B.S.: Automatic clustering of software systems using a genetic algorithm. In: Proceedings Software Technology and Engineering Practice, STEP 2099, pp. 73–81 (1999)
11. Mancoridis, S., Mitchell, B.S., Rorres, C., Chen, Y., Gansner, E.R.: Using automatic clustering to produce high-level system organizations of source code. In: Proceedings of 6th International Workshop on Program Comprehension, pp. 45–52 (1998)
12. Bavota, G., Carnevale, F., De Lucia, A., Di Penta, M., Oliveto, R.: Putting the developer in-the-loop: an interactive GA for software re-modularization. In: Fraser, G., Teixeira de Souza, J. (eds.) SSBSE 2012. LNCS, vol. 7515, pp. 75–89. Springer, Heidelberg (2012)
13. Abdeen, H., Ducasse, S., Sahraoui, H.A.: Modularization metrics: assessing package organization in legacy large object-oriented software. In: Proceedings of WCRE 2011, pp. 394–398. IEEE Computer Society Press (2011)
14. Barros, M.: An analysis of the effects of composite objectives in multi-objective software module clustering. In: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary GECCC-12, pp 1205–1212 (2012)
15. Wiggerts, T.A.: Using clustering algorithms in legacy systems remodularization. In: IEEE Working Conference on Reverse Engineering, pp. 33–43 (1997)
16. Mitchell, B.S., Mancoridis, S.: On the automatic modularization of software systems using the bunch tool. IEEE Trans. Softw. Eng. **32**(3), 193–208 (2006)
17. Anquetil, N., Denier, S., Ducasse, S., Laval, J., Pollet, D.: Software (re)modularization: fight against the structure erosion and migration preparation (2010)

18. Abreu, F.B., Goulao, M.: Coupling and cohesion as modularization drivers: are we being over-persuaded. In: Fifth European Conference on Software Maintenance and Reengineering, pp. 47–57 (2001)
19. Mkaouer, W., Kessentini, M., Shaout, A., Koligheu, P., Bechikh, S., Deb, K., Ouni, A.: Many-objective software remodularization using NSGA-III. ACM Trans. Softw. Eng. Methodol. **24**(3), 1–45 (2015)
20. Abdeen, H., Ducasse, S., Sahraoui, H., Alloui, I.: Automatic package coupling and cycle minimization. In: Proceedings of the 16th Working Conference on Reverse Engineering, pp. 103–112. IEEE CS Press, Lille, France (2009)
21. Falkenauer, E.: The grouping genetic algorithm–widening the scope of the GAs. Proc. Belg. J. Oper. Res. Stat. Comput. Sci. **33**, 79–102 (1992)
22. Falkenauer, E.: Genetic Algorithms for Grouping Problems. Wiley, New York (1998)
23. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern Recogn. **33**(9), 1455–1465 (2000)
24. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
25. Cotilla-Sanchez, E., Hines, P.D.H., Barrows, C., Blumsack, S., Patel, M.: Multi-attribute partitioning of power networks based on electrical distance. IEEE Trans. Power Syst. **28**(4), 4979–4987 (2013)
26. Hartigan, J., Wong, M.: Algorithm AS 136: a K-means clustering algorithm. J. Roy. Stat. Soc. Ser. C **28**(1), 100–108 (1979)