Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 7960610, 13 pages
https://doi.org/10.1155/2022/7960610

Research Article

An Enhanced Multifactor Multiobjective Approach for Software Modularization

Muhammad Zakir Khan,[1] **Rashid Naseem**,[2] **Aamir Anwar**,[3] **Ijaz ul-Haq**,[4] **Saddam Hussain**,[5] **Roobaea Alroobaea**,[6] **Syed Sajid Ullah**,[7] **and Fazlullah Umar**[8]

[1]*James Watt School of Engineering, University of Glasgow, Scotland, UK*
[2]*Department of Computer Science, Pak Austria Fachhochschule Institute of Applied Science and Technology, Haripur, Pakistan*
[3]*School of Computing and Engineering, University of West London, London W55RF, UK*
[4]*University of Lleida, Lleida 25003, Spain*
[5]*Department of Information Technology, Hazara University, Mansehra, Pakistan*
[6]*Department of Computer Science, College of Computers and Information Technology, Taif University, P. O. Box 11099, Taif 21944, Saudi Arabia*
[7]*Department of Information and Communication Technology, University of Agder (UiA), Kristiansand, Norway*
[8]*Department, Khana-e-Noor University, Pol-e-Mahmood Khan, Shashdarak, 1001, Kabul, Afghanistan*

Correspondence should be addressed to Saddam Hussain; saddamicup1993@gmail.com, Syed Sajid Ullah; syed.s.ullah@uia.no, and Fazlullah Umar; fazlullahumar@gmail.com

Received 16 March 2022; Accepted 4 May 2022; Published 8 June 2022

Academic Editor: Ghous Ali

Complex software systems, meant to facilitate organizations, undergo frequent upgrades that can erode the system architectures. Such erosion makes understandability and maintenance a challenging task. To this end, software modularization provides an architectural-level view that helps to understand system architecture from its source code. For modularization, nondeterministic search-based optimization uses single-factor single-objective, multifactor single-objective, and single-factor multiobjective, which have been shown to outperform deterministic approaches. The proposed MFMO approach, which uses both a heuristic (Hill Climbing and Genetic) and a meta-heuristic (nondominated sorting genetic algorithms NSGA-II and III), was evaluated using five data sets of different sizes and complexity. In comparison to leading software modularization techniques, the results show an improvement of 4.13% in Move and Join operations (MoJo, MoJoFM, and NED).

1. Introduction

Software creation, operation, and maintenance require a systemic, structured, and quantifiable approach. Software systems demand functional changes as part of their software evolution [1, 2]. To this end, an understanding of the software system is developed through its corresponding documentation [3–5]. In situations where there is no documentation or when documentation is outdated, adding new features to meet frequently changing customer requirements remains a challenging task. As a result of nonupdated systems' failure to meet requirements, complex software systems can undergo structural and quality deterioration [6]. The software system is less flexible, more difficult to understand and maintain due to its low quality [7]. To this end, approaches such as software modularization (SM) are used to solve the problems effectively [4]. According to a study published by Candela et al. [8], the analysis phase accounts for 40% to 60% of management effort. The modularization quality (MQ) metric is based on the weighted edges of the software system graph and is used to evaluate the partitioning quality. The edge weights are described in the literature using different relationships, such as direct [9, 10], indirect [11–13], and semantic similarity [11, 14]. These

methods enhance coupling and cohesion while considering a single objective with a single relationship factor or a single objective with a multifactor (MF) relationship [15].

SM is an NP-Hard problem and has been solved in the past through search-based optimization. The search-based approach, proposed by Hwa et al. [16], enhances cohesion and coupling [17, 18], and improves the software structure by enhancing the criteria for coupling and cohesion [9, 18, 19]. Moreover, meta-heuristic approaches [17, 20] are also used to solve modularization problems such as Barros [21] investigated the software clustering problem's efficiency and efficacy of using two composite objectives. An experimental investigation revealed that eliminating the composite objectives from the software clustering problem allows a multiobjective (MO) evolutionary algorithm to identify better solutions faster. Morsali and Keyvanpour [3] classified each of the techniques in this software clustering. Similarly, Srijoni et al. [9] presented SMARTKT code comments that include application-specific knowledge that matches 72% of human-annotated ground truth. The authors consider the well-known MO evolutionary algorithms (NSGA II and NSGA III) to overcome MO optimization [20]. In five different data sets, modularization based on combined features consistently outperforms modularization based on structural and nonstructural features. Furthermore, the proposed MF MO function, which includes structural and nonstructural functionalities, outperforms combined-based objective functions in leading optimization algorithms by more apparent and comprehensible modules. The results also suggest that the MO optimization strategy-based meta-heuristic algorithm outperforms other techniques. As a result, using a MF MO approach with five objectives and three relationship factors, this study provides an enhanced hybrid approach for reconstructing software systems' architectural design. Coupling and cohesiveness are considered as single objectives in TurboMQ's MO formulation. Among other things, Turbo MQ is also mentioned, as well as the five objectives of the maximizing clustering approach (MCA), roughly equal size cluster approach (RESCA), cluster cohesiveness approach (CCA), Cluster Connectedness Approach (CCoA), and intracluster connection density (ICD), as well as three MF formulations of direct, indirect, and semantic features.

The primary contribution of this research is summarized as follows:

(i) The research presented an enhanced hybrid formulation of a MO problem by considering five major quality objectives.

(ii) Proposing a new concept of MF relationships that collectively considers the direct, indirect, and semantic features.

(iii) Researchers have not investigated heuristic and meta-heuristic approaches for solving the problem of SM while considering direct, indirect, and semantic features at the same time.

The remaining sections of the paper are organised as follows: Section 2 discusses the relevant work that has been done. Section 3 describes the suggested MF, MO strategy, which is comprised of several factors and multiple objectives. Section 4 discusses the MO formulation, while Section 5 discusses the experimental setup. Section 4 discusses the MO formulation. Section 6 discusses the findings and the analysis that was done as a result of them.

2. Related Work

Over the past two decades, there has been a lot of research on the automated modularization of software systems to improve system quality by optimising the software architecture. Most SM approaches use clustering techniques [22, 23], divided into data clustering techniques and graph clustering techniques. Mkaouer et al. [24] presented a novel MO search approach using NSGA-III, which may improve package structure, decrease the number of changes, preserve semantic coherence, and reuse change history. Wen et al. [25] proposed algorithms and optimization methods for indexes. Similarity Abualigah et al. [26] presented H-KHA, a novel hybrid of the krill herd (KH) and harmony search (HS) algorithms, to improve global (diversification) searchability by increasing the number of searchable items in a collection. When a new probability component, dubbed distance, is included into the KH algorithm, the exploration searchability of krill individuals in their pursuit of the ideal global solution improves significantly. Some of these approaches have been modified for numerous purposes such as extraction of modules [16] regrouping software systems [9], extraction of functional elements [27], and so on. On the other side, SBOT has been used to expeditiously build SM and is an essential part of software system architecture. The SBOT for regrouping in terms of MQ suggested by Hwa et al. [16] and Erdemir and Buzluca [28] used it. The SBOT was enhanced, evaluated, and calibrated by few previous studies [28–30]. It is an effective way for single and multi-relationship factors (MFs), such as connectivity, artifact sharing, and semantics. The modularization problem was designed using single factor [31, 32] MO optimization problems using hill climbing (HC) and genetic algorithm (GA).

Praditwong et al. [9] improved the search-based approach by taking module cohesion and coupling under consideration and using an MO evolutionary algorithm. The new MO approach included search-based enhancements that were more effective than the SO formulation. They used the HC algorithm to address two MOs: the ECA (equal cluster size approach) and the MCA (maximize cluster approach). Later, Barros et al. [19] used a new objective to assess the efficacy of the ECA and MCA formulations. Their empirical analysis revealed that, like with MCA and ECA, equivalent results could be achieved with fewer objectives. The distinction "the gap between the maximum and a minimum number of artefacts in a module shall be reduced" was used instead of the previous one. To solve the problem, Chhabra et al. [20] used NSGA-II, which addressed four MO : PCI, PCI, IPCD, and PCI. Schmidt explored eight MOs, including standarised cumulative component dependence, subsystem relational cohesion, efferent subsystem

coupling, erent subsystem coupling, distance to subsystems, and number of forbidden outgoing-type dependence, number of package cycles, and range of subsystem compilation units, using NSGA-III. [32]. Although the search-based strategy has been broadly investigated, an effective approach is used for single and MF, that is, direct, indirect, and semantic features, and the problem of modularization is also formulated using the MO optimization problem of a single factor [20] using HC. Similarly, Huang et al. [15] proposed a novel search-based approach for grouping software modules based on various relationship factors. They argue that all existing approaches analyse the whole system as though it were a single factor, which leads to the following problems. To begin, the system's overall quality cannot be determined by a single factor: certain modules can form semantic relationships, while others can form structural ones. Second, the user of the approach should select a factor without knowing which one is the most effective.

This research examined a new approach for optimising MF MO. Researchers adopted a meta-heuristic technique to solve the search optimization problem, and we followed a search-based approach because it consistently produces better results. Owing to its significance in the early modularization of software, we used the weighting scheme for class connections to achieve this task. The usage of this concept has been proposed as a new SM mechanism for module reconstruction. To the best of my knowledge, no MF, MO approach has been given. The proposed approach outperforms prior approaches because it evaluates several connections with the same weight equal to 1 rather than only binary values. As a result of our experiment results, we were able to develop an effective and optimal SM approach.

2.1. Single-Factor Single-Objective SM. To solve the problem of SM, several researchers have used several different factors (features) in their research. Files, macros, function calls, user-defined data types, and even global access variables were used by Anquetil [33], while nonformal features included comments, identifiers, URLs, and even developer names. Artifacts were also identified as files, routines, classes, and processes.

2.1.1. Existing Single-Factor Approach. Researchers have used formal and informal, static, and dynamic relationships that are based on structural and nonstructural factors. However, the three types based on their nature are direct, indirect, and semantic. The following is a list of researchers who have investigated the connections, which are summarized in Table 1.

2.1.2. Existing Single-Objective Approach. In a single objective, only one objective is optimised.

$$F(M*) = \frac{\min}{\max F(M) \mid M \in \Psi}. \quad (1)$$

In equation (1), M^* refers to modularization, while Ψ is for the modularization feasible set. The most common problem with modularization is single-objective optimization, where F is the function of minimization or maximization. Search-based module clustering approaches are used to explore the possible partitions in the search space, and this approach is used to discover the optimal solution. Following on from a previous study that used a single-factor formulation known as MQ [16] to reveal better solutions throughout the search, we have concentrated on TurboMQ presented as one of the internal metrics that has been used in many research papers to evaluate the quality of recovered architectures as indicated in equation (2) to reveal better solutions throughout the search. The Turbo MQ measurement was created in order to overcome the two limitations of the Basic MQ measurement. Turbo MQ is significantly faster than Basic MQ and supports multidimensional graphs with edge weights (computational complexity is $O(V)$).For an MDG partitioned into k clusters, the Turbo MQ measurement is obtained by multiplying the Cluster Factor (CF) for each cluster by the Turbo MQ measurement.

$$\text{TurboMQ} = \sum_{i=1}^{k} CFi,$$

$$CF = \begin{cases} 0, & \text{if } \mu i = 0, \\ \dfrac{2\mu}{2\mu + \epsilon}, & \\ & \text{otherwise.} \end{cases} \quad (2)$$

Cluster factor (CF_k) is the sum of its modules. Each CF_k measures the ratio of intra (u) and interedge (ε) Ck weight sum. Researchers also employ additional objective functions in their studies. The objective function is summarized in Table 2.

2.2. MF MO SM. The software system can be modelled as a graph, with nodes representing classes and edges representing relationships between them. A metric called MQ is calculated across the weighted edges of the software system's graph representation to measure the quality of a given clustering partitioning problem. A parameter that describes the "relationship" between modules is known as an issue parameter. The edge weights are described in the literature using several different relationship factors: Direct [9, 38], indirect [30], and semantic similarity [39] were extensively studied. In addition, details such as changing history [39], physical locations of modules [40, 41], and design evolution features [42] were considered.

2.2.1. Existing Single-Objective Approach. The parameterized variant of single-factor SM is MF. MF allows each cluster of nodes or each node-to-node edge to have different weights depending on different relationship criteria, resulting in a clustering that incorporates multiple aspects of

Table 1: Work-related to direct, indirect, and semantic relationship.

S. No.	Papers	Entity	Direct	Indirect	Semantic
01	[33]	Class	Method call	Not used	Comments
02	[34]	Class	Method call, references, and extends	Not used	Not used
03	[31]	Class	Member function	Not used	Members variable names, class, and function names

Table 2: Work-related to single-objective function.

S. No.	Paper	Fitness function
01	[10]	MQ, turbo MQ
02	[35]	Entropy-based objective function
03	[36]	Entropy-based objective function
04	[4]	Cohesion and coupling
05	[37]	MQ

module relationships. Hwa conducted a study on MF and presented the MF module clustering (MFMC) formulation in their analysis [15]. They modified the SF formula to create two MF-focused search-based approaches, which they then applied and evaluated with the HC algorithm. The results of the empirical evaluation reveal that formulations of MFMC yield modularization that are on average 10.69% more comparable than SF formulations. Different edges are assigned different weights according to their relationships, resulting in a cluster with several module clustering features. MF relationships between modules are represented by the number of modules (n) and the number of MF relationships (m), that is, $G = (N, E)$. Each edge is represented by E as follows by Huang et al. [15].

$$\begin{aligned} e &= (n_a, n_b, W), n_a, n_b \in N, a \neq b, \\ W &= W_{ab}(1), W_{ab}(2), \ldots, {}_{ab}(n). \end{aligned} \tag{3}$$

Based on different types of relationships, W_{ab} is the weight of the edges between n_a and n_b. When the edge weights are added together, the strength of the connections between the two edges is revealed.

2.2.2. Existing MO Approach. MOs optimise more than one objective. Each of the existing approaches merges the twin objectives of cohesion and coupling into a single target feature to avoid aggravating the suboptimal solution result.

$$\begin{aligned} F(M^*) = \min(&F1(M), F2(M)), \ldots, \\ &Fm(M) \mid M \in \psi. \end{aligned} \tag{4}$$

The target function is represented by F, and the total number of targets is represented by m. Praditwong et al. [9] provide a novel method for applying Pareto front optimality to an MO issue (the set of all nondominated solutions in an objective space). Scalable modularization solutions are among the MO modularization approaches that provide developers additional options to select from based on their requirements. The optimal Pareto scale integrates a variety of dimensions into a single common metric scale.

Our study aim is to employ MO module optimization, which involves using multiobjective evolutionary algorithms (MOEAs) to optimise many objectives at the same time. Instead of using the term MO, we chose MF to emphasise that many factors are weighted during actual cluster formulation. Unlike Praditwong et al. [9], this research aims to develop new approaches that incorporate multiple elements and factors to get a single solution. Table 3 shows the work related to MOs.

3. Proposed MF MO Approach

Numerous approaches have been used to various factors that degrade the structure and results of SM. Taking the call dependency graph (CDG) as a direct, the majority of them were only used for modularization (structure features).There are not enough tools to extract indirect and semantic features. As a result, the proposed approach considered both direct and indirect features, as well as semantic features. The recommended method involves using an MF, MO evolutionary formulation approach with five objectives and three relationship factors. MQ and MMQ are MO formulations that combine the two objectives of minimal coupling and high cohesion into a single objective. We considered BasicMQ and TurboMQ, as well as the five objectives and three relationship factors:

(1) Maximizing clustering approach (MCA)

(2) Cluster connectedness approach (CCoA)

(3) Roughly equal size cluster approach (RESCA)

(4) Cluster cohesiveness approach (CCA)

(5) Intracluster connection density (ICD)

The three factors formulation includes

(1) Direct (connectivity)

(2) Indirect (artifact sharing)

(3) Semantic

The proposed hybrid MF with MO formulation considers both structural (direct and indirect) and nonstructural (semantic) relationship features at the same time. The structural characteristics investigated (Global variable, Macros, Overriding, and Class containment) include indirect features (Inheritance, Function Calling, Class Calling, Interface, Class lies inner, and Static Inner Class Calling) and Calling Dependency (Inheritance, Function Calling, Class Calling, Interface, Class lies inner, and Static Inner Class Calling).The similarity between identifiers and comments is one of the nonstructural features considered. Figure 1 depicts the whole process of the proposed enhanced hybrid MF MO. The general structure of the proposed methodology is given in Figure 2. The following is the suggested strategy: First, an object-oriented software

Table 3: Work related to multiobjectives.

S. No.	Paper	Fitness function
01	[75]	Class inter and intramodule change coupling, index module counts, and size
02	[43]	FUP-based cohesion and semantic relatedness
03	[44]	The cohesion of relational subsystems (maximize), for composite components, there is a standard subsystem dependency (converge to 1.0), coupling error in a subsystem (minimize), subsystem coupling (minimize), distances between subsystems (minimize), restrict the number of outbound-type dependencies (minimize), and the number of cycles on packages (minimize), subsystem compilation units (minimize)
04	[34]	Intra and intercluster dependency, cluster count, and module count per cluster
05	[20]	Package connectivity index, density of intrapackage connections, and package size index
06	[20]	minimize modifications to the package's structure, maintain semblance coherence reuse affects history
07	[45]	Package structure, minimum change, MCA, and ECA

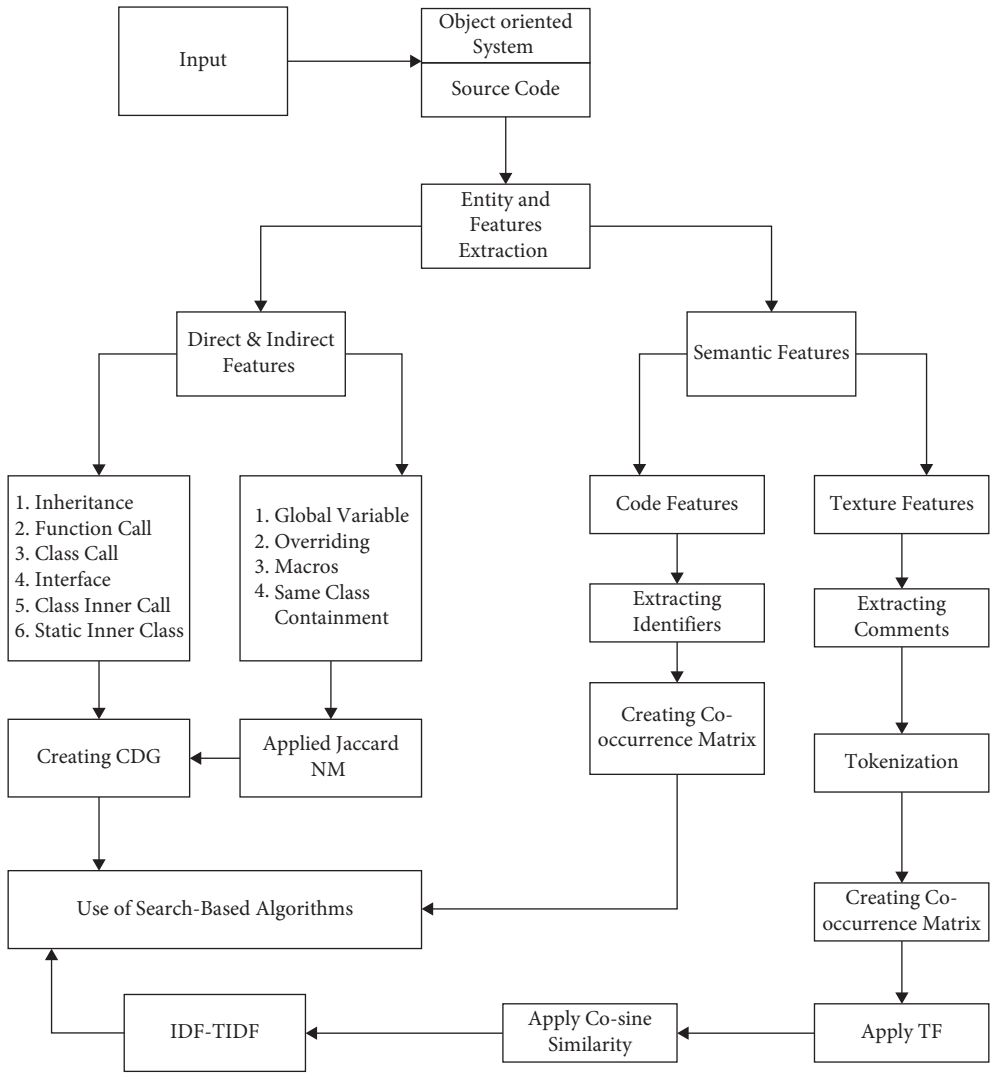

Figure 1: Process of Structural and Nonstructural feature extraction.

system is used to extract entities (classes) and relationships (direct, indirect, and semantic). Second, assigning weights to different relationships (in this paper, we used weight 1), then aggregating the weights and allocating them to the relationship, shows their strength. In the third phase, a Weight Class Connection Graph (WCCG) is constructed based on the relationships to represent the software system, and then MFMO criteria are defined, and an Evolutionary Algorithm is used to them to provide the software system's output.

3.1. Entity and Relationships Extraction. Classes are the building blocks of OOPS that encapsulate an entity's

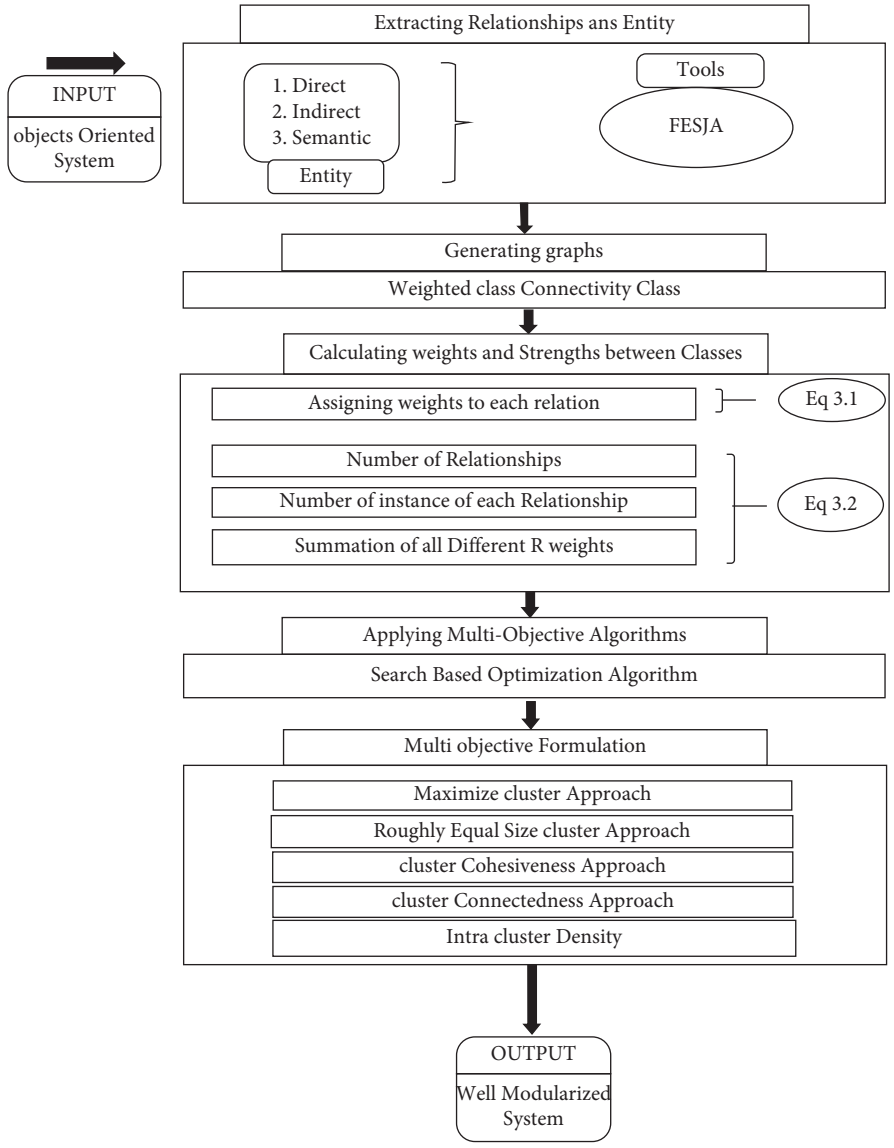

Figure 2: Proposed approach schematic diagram.

properties and functions. According to the [46] reconstruction of the architecture, a class is vital in software. It is an essential part of object-oriented software. The smallest, architecturally significant elements are entities [47]. They participate in the clustering phase of the automated software clustering and modularization process and become cluster participants [37]. Although these classes are linked by structural, dynamic, static, semantic, and conceptual relationships [9].Our research focused on direct, indirect, and semantic connections. Fact Extractor System for Java Software and Fact Extractor System for Java Software were used in this study (FESJA). Using the FESJA to extract the relationships, which are described briefly as follows:

3.2. Direct Relationships. The researchers [20, 27] have considered some direct relationships that will represent the system in its true meaning.

(1) *Inheritance (I).* Access to all of class A's methods and attributes is denoted by class B.

(2) *Function Call (FC).* Container class B invokes at least one method from container class A.

(3) *Class Call (CC).* Class A contains a class B object.

(4) *Interface (IBI).* Class A inherits an interface's abstract methods when it implements.

(5) *Class Lies Inner (CLI).* Entity-to-object connection in which a function parameter is an entity A.

(6) *Static Inner Class (SIC).* Class A has access to the private static data of members of outer class B.

3.3. Indirect Relationships. The following are some indirect relationships that researchers frequently use to depict the system. The description of extracting indirect relationships is given. The whole process is explained in Figure 3.

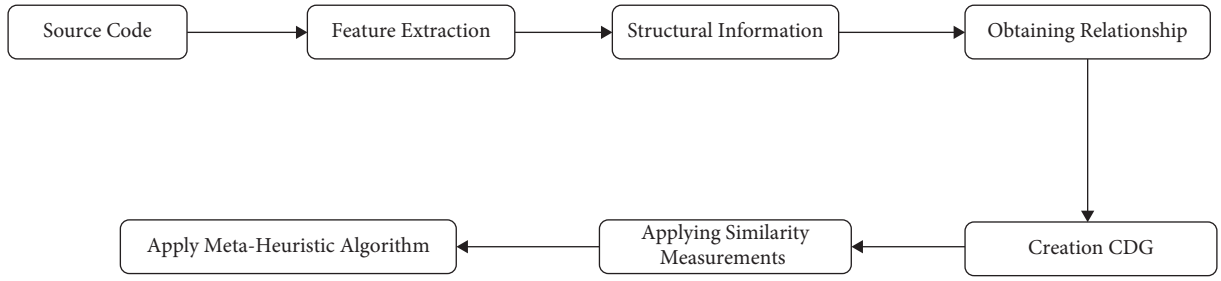

Figure 3: Proposed approach indirect relationship extraction.

(1) *Global Variable (Gv)*. When Class A and Class B share a global variable.
(2) *Overriding (O)*. Class B and C have access to the parent class.
(3) *Macros (Mc)*. Class A and B macros are the same.
(4) *Same Class Containment (SCC)*. When two classes share objects.

3.4. Semantic Relationships. Similarities between comments and extracted identifiers are considered semantic characteristics for SM:

(1) Maximize similarity between comments
(2) Identifier name similarity should be maximized
(3) Maximize the existing inheritance relationship between two identifiers
(4) If a call relation exists, it should be maximized inside the module and minimized across it

Figure 4 shows the entire process. This study used synthetic features like function calls, variable calls, and inheritance. It appears to be a better SM, but we could potentially get a better SM by extracting from identifiers, comments, and other nonsynthetic characteristics in software systems' source code. Misra [11] proposed combining synthetic and nonsynthetic features to remodularize the software system. The comments and identifiers are taken from the source code. This work used Jaccard-NM instead of Jaccard after extracting direct relationships since Jaccard-NM produces better results and eliminates the random decision after two similarities [48].

The feature metric data table $N * P$, where N denotes the entity and P denotes the features, can also be analyzed. It is a MoJo (Move and Join (MoJo) Operation)-based evaluation metric that can be used to test the stability of two modularizations and calculate their distance. Instead, a low MoJo score indicates that two partitions are comparable. These steps are specified [47, 49].

3.5. Assigning of Weights. Numerous quality criteria, including cohesion and coupling, are used to evaluate the system's efficiency, ensuring that the system's categories are coupled in such a way that a good SM is produced. Since it is based on basic forms of relationships, instances, and cumulative weights between classes, the relationship between classes is complicated. The connections will be assigned weights both internally and externally as a result of this research. Following the weighting formula [20], the weights of each relationship are added together to produce an aggregate sum of all contributing relationships in this study.

$$Wk = \sum_{i=1}^{n} \sum_{j \in Ni} .Nk(Ci, Cj) + \sum_{i=1}^{n} \sum_{j \notin Ni} .Nk(Ci, Cj),$$

$$W = Wab(1), Wab(2), \ldots, Wab(n). \quad (5)$$

In equation (5), C stands for Classes. Classes Ci and Cj have N_k instances between them, while classes with the same cluster have Ni instances. The W_{ab} is a na/nb edge-weighted average based on relationships. It is revealed by adding the weights of the two edges.

4. MO Formulation

This study considered the MCA, RESCA, CCA, CCoA, and IICD Approach.

4.1. Maximizing Clustering Approach (MCA). The following set of objectives is used by MCA:

(1) The number of intraedge clusters should be increased
(2) To minimize the overall number of interedge clusters
(3) The number of clusters should be increased
(4) The MQ value should be maximized
(5) To reduce the number of separate clusters

Maximizing clusters, which is uncommon in SM, eliminates isolated clusters. To expand the number of clusters, not all modules inside a cluster need to be concise. More clusters in a system means more advantages from modularization [9].

4.2. Roughly Equal Size Cluster Approach (RESCA). A modular structure with roughly equal-sized clusters is produced in ECA, which helps in cluster disordering. It prevents large clusters and isolated clusters [9, 19]. Only one objective is different between MCA and MCA: the number of modules in a cluster.

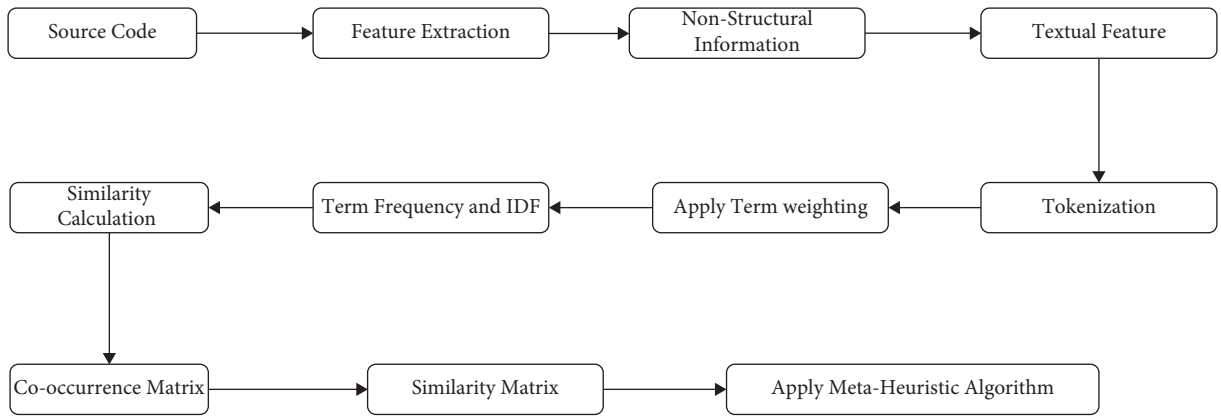

Figure 4: Proposed approach of semantic relationship extraction.

4.3. Cluster Cohesiveness Approach (CCA). The CCA measures how closely related artifacts in a cluster or node are. The cluster's intraedge connectivity is examined [9].

$$CCA(Ci, Cj) = \sum_{Ri}^{Rn} \sum_{j \in Ni} . \left(\sum_{Ri}^{Rn} WkNk(Ci, Cj) \right), \quad (6)$$

where $\sum_{Ri}^{Rn} \sum_{j \in Ni}$, denotes the total number of relationships in the given cluster.

4.4. Cluster Connectedness Approach (CCoA). Using the CCoA, you may determine how well objects in various clusters are connected to one another,

$$CCoA(Ci, Cj) = \sum_{Ri}^{Rn} \sum_{j \neq Ni} . \sum_{Ri}^{Rn} WkNk(Ci\,Cj), \quad (7)$$

where $\sum_{Ri}^{Rn} \sum_{j \neq Ni}$, represent the total of all nonclustered connections.

4.5. Intracluster Density Approach (ICDA). The optimum cluster size has almost equal numbers of artefact distributions inside clusters. However, this is not always feasible because when creating random clusters based on similarity and dissimilarities, it tends to divert to one side. Skewness in artefact distribution within clusters will be avoided by using the cluster size index (CSI). To cope with such a problem, we used ICD, which is defined as follows:

$$\text{intracluster connection density (ICD)} = \frac{C_{\min}}{C_{\max}}. \quad (8)$$

The minimum and maximum number of classes in a cluster is $C_{\min}$ and $C_{\max}$. The value decreases as the cluster size is larger, whereas it increases as the cluster size gets smaller.

5. Experimental Setup

To assist the MFMO in producing high-quality SM results in the form of an optimised solution through the use of SM techniques. The experimental setup includes (1) a description of the software system, (2) data collection, (3) multiple criteria for evaluating the results, and (4) search-based modularization approaches.

5.1. Software System Description. This paper aims to develop object-oriented software systems with a reasonable number of clusters and lines of code (LOCs). These five databases (software systems) were selected for their varying sizes and program complexity. Table 4 shows the description of the software system (data set).

5.2. Collection of Results. Approaches based on search are contentious because they work on the same chromosomes or, in some cases, on many runs at the same time. Owing to the stochastic nature of SBOT, we must collect data for each test programme (a total of 30 occasions). In an MO algorithm, we collect nondominated sorted solutions, while for a single objective, we pick the best-dominated iteration.

5.3. Evaluation Results Criteria. In this study, the modularization of heuristic and meta-heuristic algorithms was examined using five Java software systems, which were evaluated using two fundamental approaches: internal criteria and external criteria. The internal characteristics of the resulting modularization are evaluated by an internal assessment evaluation. MQ [16], cohesion and coupling [21], and the number of clusters and cluster size [50] are only a few of the quality characteristics for an internal modularized system. The focus of this study is TurboMQ, a popular research internal evaluation tool. The second evaluation is external, and its objective is to analyse and describe the degree of similarity between the achieved modularization and the expert-produced modularization (the software system's or developer's lone author) for resembling as much as possible as defined by Schmidt et al. [33]. For external, we compared the modularization provided by the algorithms with the expert decomposition using MoJo [43, 47] and MoJoFM [27, 47].

Table 4: Description of the software system (data set).

System name	sLOC	Blank lines	LOC	Classes	Function	Variable
Bash master	32,979	8,279	41,258	239	847	2,372
Bunch-master	21,000	4,099	25,099	166	1,144	2,047
PMD-master	10,642	3,240	13,882	290	1,093	407
NekoHtml Master	6,871	1,474	8,345	48	393	659
Servlet-master	2,614	1,488	4,102	27	247	230
Bash master	32,979	8,279	41,258	239	847	2,372

5.3.1. MoJo and MoJoFM. It is necessary to migrate from one modularized system to another expert-dissect system in order to use the MoJo method. It is referred to as a distance criteria because the MoJo method is used when the minimum number of MoJo steps required to move from a modularized system to a decomposed expert system is less than the number of steps required to move from a decomposed expert system. It is a distance criterion because the modularized and expert decomposed systems become more similar as the number of MoJo steps lowers. This is how it is described:

$$\text{MoJoFM} = 1 - \frac{mno(A, B)}{\max(mno(VA, B))}. \tag{9}$$

The least number of steps required to convert modularization system A to B is denoted by mno (A, B), while the maximum number of the lowest steps required for MoJo to convert A to B is denoted by max (mno (A, B)). The fundamental difference between MoJo and MoJoFM is that MoJoFM requires expert decomposition, which is the primary difference between the two. It is more likely that the modularized system will more closely resemble the expert-constructed system if the MoJoFM values increase and the MoJo value decreases.

It should be noted that contacting the developer of a software system that is about to be reviewed is difficult due to the developer's busy schedule or the risk of quitting the company. However, as countless academics have demonstrated [27, 47], there is always a middle ground that should be pursued. To cope with this scenario, we have a few options.

(i) Identify the module and the number of entities (here, classes) that are associated with it

(ii) Validate the existing module's source code with comments

(iii) Modules with less than five classes should be combined

(iv) Software development expertise was enlisted

5.3.2. Nonextreme Distribution (NED). According to Alswaitti et al. [27], a good modularization system has module sizes that are neither too large nor too small. The modularized software system, on the other hand, has a well-balanced class distribution in each module. The two conditions should be avoided by an algorithm when dealing with nonextreme distributions (NED). (i) Some files belong to one of a few large clusters or are classes within one of them (black holes). (ii) The majority of clusters are singletons (dust clouds). The NED provided by Prajapati and Chhabra [43] and Rahman et al. [51] to examine the extreme distribution of module size is given below, and it can be found by following the instructions.

$$\text{NED} = \frac{\sum_{i=1}^{n} Mi, NED|Mi|}{t}, \tag{10}$$

where Mi is NED if $5\langle|Mi|\rangle 100$.

The number of modules and the objective system are shown in equation (10). The solutions with the highest NED values are the most suitable and stable. They consider cluster sizes of less than 5 to more than 100 to be excessive. According to the definition, "the ratio of the number of files in the nonextreme cluster to the software's total number of target source data" is the ratio described. The better the module class distribution, the better the NED value.

6. Results and Analysis

The comparison of relationships across algorithms and algorithm-based comparison have been discussed in this section.

6.1. Comparison of Relationship across Algorithms. This section contrasts direct and indirect, semantic, and combined relationships. Three external evaluations, include MoJoFM (should be high), MoJo (should be minimal), and NED; centred on these relationships (should be maximum), Table 5 compares algorithmic relationships.

6.2. Relationships Comparison Using Algorithms. On the basis of three data sets, this section examines the effects of GA, the NSGA II, and the HC on direct, indirect, semantic, and combined relations in each data set. In Table 6, the letters R1 and R2 represent direct and indirect relationships, R3 represents semantic relationships, and R4 represents combining relationships. Using MoJoFM, we can compare three different relationship factors across three different algorithms. The higher the value of MoJoFM, the more the system will be similar to the expert system. The total number of counts for GA is 5, NSGA is 8, and HC is 2. The count shows the dominance of the NSGA, which means MO performance. The NSGA shows better results on five data

Table 5: Comparison of relationship across algorithms.

Algorithms	Results of external evaluation of bash data set								
	Direct indirect			Semantic			Combined		
	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED
GA	83.09	39.73	96.40	83.20	39.46	96.68	83.54	38.67	97.40
NSGA	83.35	39.13	97.00	83.35	39.13	96.82	83.38	39.13	96.79
HC	83.42	38.43	97.26	83.29	39.26	96.91	83.32	39.20	97.37
Average	83.28	39.09	96.88	83.28	39.28	96.80	83.41	39.00	97.18
Algorithms	Results of external evaluation of bunch data set								
	Direct indirect			Semantic			Combined		
	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED
GA	66.54	53.53	94.57	65.00	56.00	94.57	65.90	54.56	94.97
NSGA	66.47	53.43	96.28	66.64	54.7	96.40	66.00	54.4	95.72
HC	65.87	54.6	95.38	65.67	54.8	95.34	65.92	54.47	95.18
Average	66.29	53.85	95.41	65.77	55.16	95.43	65.94	54.47	95.29
Algorithms	Results of external evaluation of NekoHTML data set								
	Direct indirect			Semantic			Combined		
	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED
GA	25.04	28.10	81.25	25.76	25.57	83.10	23.09	30.06	81.25
NSGA	25.37	28.10	85.00	24.95	28.70	84.02	23.82	28.90	85.62
HC	22.52	28.6	85.27	25.28	28.53	83.05	23.47	29.13	84.02
Average	24.31	28.26	83.84	25.33	27.60	83.39	23.46	29.36	83.63
Algorithms	Results of external evaluation of PMD data set								
	Direct indirect			Semantic			Combined		
	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED
GA	31.65	16.88	99.00	31.01	16.58	96.89	31.45	16.78	96.94
NSGA	31.71	16.36	98.35	31.48	17.01	98.34	31.74	16.32	97.83
HC	31.51	16.92	97.35	31.63	16.94	97.27	31.67	16.83	97.27
Average	31.62	16.72	98.23	31.37	16.84	97.5	31.62	16.64	97.34
Algorithms	Results of external evaluation of servlet data set								
	Direct indirect			Semantic			Combined		
	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED	MoJoFM	MoJo	NED
GA	32.61	15.40	66.67	37.25	14.94	90.86	38.84	13.93	66.67
NSGA	35.26	14.80	67.53	37.10	14.03	88.39	37.38	14.10	68.88
HC	33.47	14.93	70.00	34.49	14.83	71.48	37.52	14.10	69.50
Average	33.78	15.04	68.06	36.28	14.60	83.57	37.91	14.04	68.35

Table 6: Evaluation of algorithms on MoJoFM.

Algorithm	Results of external evaluation with respect to MoJoFM								
	Bash			Bunch			NekoHTML		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
GA	83.09	83.20	83.54	66.54	65.00	65.90	25.04	25.76	23.09
NSGA	83.35	83.35	83.38	66.47	66.64	66.00	25.37	24.95	23.82
HC	83.42	83.29	83.32	65.87	65.67	65.92	22.52	25.28	23.47
	PMD				Servlet				
	R1	R2	R3		R1	R2	R3		
GA	31.65	31.01	31.45		32.61	37.25	38.84		
NSGA	31.71	31.48	31.74		35.26	37.10	37.38		
HC	31.51	31.63	31.67		33.47	34.49	37.52		
Total count	GA			NSGA			HC		
	5			8			2		

sets with respect to the MoJoFM evaluation metric. Table 7 represents the comparison of three relationship factors with respect to three algorithms based on MoJo. The lower the value of MoJo, the more the system will be similar to the expert system. In Table 7, the total number of counts for GA is 2, NSGA is 12, and HC is 1. The count shows the dominance of the NSGA, which means MO performance. The NSGA shows better results on five data sets with respect to the MoJoFM evaluation metric.

Table 8 represents the comparison of three algorithms across NED values that are higher than the NED value, more like the original system. The total number of counts for GA is

Table 7: Evaluation of algorithms on MoJo.

Algorithm	Results of external evaluation with respect to MoJo								
	Bash			Bunch			NekoHTML		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
GA	39.73	39.46	38.67	53.53	56.00	54.56	28.1	25.57	30.06
NSGA	39.13	39.13	39.13	53.43	54.7	54.4	28.1	28.7	28.9
HC	38.43	39.26	39.20	54.60	54.8	54.47	28.6	28.53	29.13
	PMD				Servlet				
	R1	R2	R3		R1	R2	R3		
GA	16.88	16.58	16.78		15.40	14.94	13.93		
NSGA	16.36	17.01	16.32		14.80	14.03	14.10		
HC	16.92	16.94	16.83		14.93	14.83	14.11		
Total count	GA		NSGA			HC			
	2		12			1			

Table 8: Evaluation of algorithms on NED.

Algorithm	Results of external evaluation with respect to NED								
	Bash			Bunch			NekoHTML		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
GA	96.40	96.68	97.40	94.57	94.57	94.97	81.25	83.10	81.25
NSGA	97.00	96.82	96.79	96.28	96.40	95.72	85.0	**84.02**	**85.62**
HC	97.26	96.91	97.37	95.38	95.34	95.18	85.27	83.05	84.02
	PMD				Servlet				
	R1	R2	R3		R1	R2	R3		
GA	99.00	96.89	96.94		66.67	90.86	66.67		
NSGA	98.35	98.34	97.83		67.53	88.39	68.88		
HC	97.35	97.27	97.27		70.00	71.48	69.50		
Total count	GA		NSGA			HC			
	3		7			5			

3, NSGA is 7, and HC is 5. The NSGA shows better results on five data sets with respect to the NED evaluation metric.

7. Discussion and Conclusion

The concept of computing information-theoretical similarity is uncommon in search-based software engineering (SBSE). SBSE experts will not use the information-theoretical similarity measure when it comes to SM. Rather than focusing on how to evaluate structural and semantic similarity, this study looked at how to improve the hybrid idea of mixed relationships by combining structural and nonstructural similarity into a single platform to modularize the software system. Furthermore, five (or more) objectives are optimised and used at the same time. As a result, an MO meta-heuristic algorithm based on MF relationships is a feasible alternative.

Tables 5 to 8 show the results of five data sets using three different approaches and three different relationships. Since data sets differ in size and complexity, three techniques, MoJo, MoJoFM, and NED, show different responses on five data sets. To begin with, the three behaviours of the algorithms are distinct in the Bash data set, indicating that GA performs better on R3, which is a combined relationship, while NSGA performs better on R1 (Direct-Indirect Relationships), R2 (Semantic Relationships), and HC performs better on R1 (Direct-Indirect Relationships). The behaviour of these three algorithms is the contrary. Because there are no direct-indirect interactions between classes in the Bash data set, and because comments (semantic behaviour) are absent from the source code, cohesion and coupling are minimal in the Bash data set. The three algorithms also diverge from the Bunch data set. NSGA outperforms GA and HC in three relationships. The direct and indirect relationships are reasonable in the source code; however, comments appear in every class. On the NekoHTML data set, the NSGA surpasses the NSGA on Direct Indirect and Combined, except for Semantic. The three algorithms are also not the same as the Bunch data set. On three relationships, GA and HC report poor results. However, NSGA provides better results. Despite the fact that the direct-indirect relationships in the source code are reasonable, comments appear in each class. However, when it comes to the NekoHTML data set, the NSGA once again outperforms the HC on Direct Indirect and Combined, with the exception of Semantic, where relations between classes are fair due to the absence of correlations inside classes. Except for semantic, NSGA performs better on direct indirect and combined in the PMD data set, where the direct-indirect relationship is satisfactory, but class comments are zero (mostly empty). This is because the data set has almost no comment relationships and few direct-indirect relationships in classes, and HC beats GA and HC in all three relationships.

In conclusion, since NSGA is a more refined variant of GA, it produces better outcomes than HC and GA. As a result of greed, HC shows no reasonable result. Table 5

shows relationship comparisons based on external evaluations using MoJoFM, MoJo, and NED. Apart from NekoHTML, where the source code has semantic relationships that produce better results due to semantic cohesiveness among the classes, combined relationships perform better in the Bash, Bunch, PMD, and Servlet data sets. We concluded that NSGA outperforms other algorithms, whereas SM benefits from combining relationship features. Our MFMO approach has been completely demonstrated by the beneficiaries of this enhanced hybrid approach. In addition, five objective functions are optimised and used at the same time. As a result, finding an MO metaheuristic algorithm with MF relationships for improved SM is a plausible choice.

Data Availability

The data used in this research can be obtained from the corresponding authors upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors are grateful to the Taif University Researchers Supporting Project number (TURSP-2020/36), Taif University, Taif, Saudi Arabia.

References

[1] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software development in startup companies: the greenfield startup model," *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 585–604, 2016.

[2] F. Shah, A. Anwar, H. AlSalman, S. Hussain, and S. Al-Hadhrami, "Artificial Intelligence as a Service for Immoral Content Detection and Eradication," *Scientific Programming*, vol. 2022, Article ID 6825228, 9 pages, 2022.

[3] F. Morsali and M. R. Keyvanpour, "Search-based software module clustering techniques: a review article," in *Proceedings of the 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pp. 0977–0983, IEEE, Tehran, Iran, December 2017.

[4] M. N. Adnan, M. R. Islam, and S. Hossain, "Clustering software systems to identify subsystem structures using knowledgebase," in *Proceedings of the 2011 Malaysian Conference in Software Engineering*, pp. 445–450, IEEE, Johor Bahru, December, 2011.

[5] W. Xu, J. Zhang, Y. Yuan, X. Wang, Y. Liu, and M. I. Khalid, "Towards efficient verifiable multi-keyword search over encrypted data based on blockchain," *PeerJ Computer Science*, vol. 8, p. e930, 2022.

[6] S. Ducasse and D. Pollet, "Software architecture reconstruction: a process-oriented taxonomy," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 573–591, 2009.

[7] I. U. Rehman, D. Sobnath, M. M. Nasralla et al., "Features of mobile apps for people with autism in a post covid-19 scenario: current status and recommendations for apps using AI," *Diagnostics*, vol. 11, no. 10, p. 1923, 2021.

[8] I. Candela, G. Bavota, B. Russo, and R. Oliveto, "Using cohesion and coupling for software remodularization," *ACM Transactions on Software Engineering and Methodology*, vol. 25, no. 3, pp. 1–28, 2016.

[9] K. Praditwong, M. Harman, and X. Yao, "Software module clustering as a multi-objective search problem," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 264–282, 2011.

[10] S. Majumdar, S. Papdeja, P. P. Das, and S. K. Ghosh, "Smartkt: a search framework to assist program comprehension using smart knowledge transfer," in *Proceedings of the 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, pp. 97–108, IEEE, ofia, Bulgaria, July, 2019.

[11] M. Z. Khan, R. Naseem, A. Anwar et al., "A novel approach to automate complex software modularization using a fact extraction system," *Journal of Mathematics*, vol. 2022, Article ID 8640596, 19 pages, 2022.

[12] J. Misra, K. Annervaz, V. Kaulgud, S. Sengupta, and G. Titus, "Software clustering: unifying syntactic and semantic features," in *Proceedings of the 2012 19th Working Conference on Reverse Engineering*, pp. 113–122, IEEE, Kingston, ON, Canada, October, 2012.

[13] Y.-S. Seo and J.-H. Huh, "Gui-based software modularization through module clustering in edge computing based iot environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 3, pp. 1625–1639, 2019.

[14] C. Cho, K.-S. Lee, M. Lee, and C.-G. Lee, "Software architecture module-view recovery using cluster ensembles," *IEEE Access*, vol. 7, Article ID 72872, 2019.

[15] Y. Huang, S. Huang, H. Chen et al., "Towards automatically generating block comments for code snippets," *Information and Software Technology*, vol. 127, Article ID 106373, 2020.

[16] J. Hwa, S. Yoo, Y.-S. Seo, and D.-H. Bae, "Search-based approaches for software module clustering based on multiple relationship factors," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 07, pp. 1033–1062, 2017.

[17] B. S. Mitchell and S. Mancoridis, "On the evaluation of the bunch search-based software modularization algorithm," *Soft Computing*, vol. 12, no. 1, pp. 77–93, 2007.

[18] M. I. Khalid, J. Iqbal, A. Alturki, S. Hussain, A. Alabrah, and S. S. Ullah, "Blockchain-Based Land Registration System: A Conceptual Framework," *Applied Bionics and Biomechanics*, vol. 2022, Article ID 3859629, 21 pages, 2022.

[19] Y. Fu, Z. Lei, S. Cai, J. Lin, and H. Wang, "Wca: a weighting local search for constrained combinatorial test optimization," *Information and Software Technology*, vol. 122, Article ID 106288, 288 pages, 2020.

[20] J. K. Chhabra, "Improving package structure of object-oriented software using multi-objective optimization and weighted class connections," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 3, pp. 349–364, 2017.

[21] M. d. O. Barros, "An analysis of the effects of composite objectives in multi-objective software module clustering," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, p. 1205, Philadelphia Pennsylvania USA, July, 2012.

[22] L. Chang, W. Li, L. Qin, W. Zhang, and S. Yang, "$\mathsf {pSCAN}$: fast and exact structural graph clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 387–401, 2017.

[23] S. Liu, B. Zhou, D. Huang, and L. Shen, "Clustering mixed data by fast search and find of density peaks," *Mathematical Problems in Engineering*, vol. 2017, Article ID 5060842, 7 pages, 2017.

[24] W. Mkaouer, M. Kessentini, A. Shaout et al., "Many-objective software remodularization using nsga-iii," *ACM Transactions on Software Engineering and Methodology*, vol. 24, no. 3, pp. 1–45, 2015.

[25] D. Wen, L. Qin, Y. Zhang, L. Chang, and X. Lin, "Efficient structural graph clustering," *Proceedings of the VLDB Endowment*, vol. 11, no. 3, pp. 243–255, 2017.

[26] L. M. Abualigah, A. T. Khader, E. S. Hanandeh, and A. H. Gandomi, "A novel hybridization strategy for krill herd algorithm applied to clustering techniques," *Applied Soft Computing*, vol. 60, pp. 423–435, 2017.

[27] M. Alswaitti, M. Albughdadi, and N. A. M. Isa, "Density-based particle swarm optimization algorithm for data clustering," *Expert Systems with Applications*, vol. 91, pp. 170–186, 2018.

[28] U. Erdemir and F. Buzluca, "A learning-based module extraction method for object-oriented systems," *Journal of Systems and Software*, vol. 97, pp. 156–177, 2014.

[29] A. C. Kumari and K. Srinivas, "Hyper-heuristic approach for multi-objective software module clustering," *Journal of Systems and Software*, vol. 117, pp. 384–401, 2016.

[30] C. C. Venters, R. Capilla, S. Betz et al., "Software sustainability: research and practice from a software architecture viewpoint," *Journal of Systems and Software*, vol. 138, pp. 174–188, 2018.

[31] L. Mu and C. K. Kwong, "A multi-objective optimization model of component selection in enterprise information system integration," *Computers & Industrial Engineering*, vol. 115, pp. 278–289, 2018.

[32] J. K. Chhabra, "Harmony search based remodularization for object-oriented software systems," *Computer Languages, Systems and Structures*, vol. 47, pp. 153–169, 2017.

[33] F. Schmidt, S. MacDonell, and A. M. Connor, "Multi-objective reconstruction of software architecture," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 06, pp. 869–892, 2018.

[34] A. Prajapati and J. K. Chhabra, "A particle swarm optimization-based heuristic for software module clustering problem," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 7083–7094, 2018.

[35] H. Izadkhah and M. Tajgardan, "Information theoretic objective function for genetic software clustering," *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 46, p. 18, 2019.

[36] M. Kargar, A. Isazadeh, and H. Izadkhah, "Semantic-based software clustering using hill climbing," in *Proceedings of the 2017 International Symposium on Computer Science and Software Engineering Conference (CSSE)*, pp. 55–60, IEEE, Shiraz, Iran, October, 2017.

[37] S. Muhammad, O. Maqbool, and A. Q. Abbasi, "Evaluating relationship categories for clustering object-oriented software systems," *IET Software*, vol. 6, no. 3, pp. 260–274, 2012.

[38] R. Terra, M. T. Valente, S. Miranda, and V. Sales, "Jmove: a novel heuristic and tool to detect move method refactoring opportunities," *Journal of Systems and Software*, vol. 138, pp. 19–36, 2018.

[39] Y. Liang and K. Zhu, "April Automatic generation of text descriptive comments for code blocks Proceedings of the AAAI Conference on Artificial Intelligence," vol. 32, no. 1, 2018.

[40] A. Ouni, M. Kessentini, H. Sahraoui, K. Inoue, and M. S. Hamdi, "Improving multi-objective code-smells correction using development history," *Journal of Systems and Software*, vol. 105, pp. 18–39, 2015.

[41] L. Xiao, Y. Cai, and R. Kazman, "Design rule spaces: a new form of architecture insight," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 967–977, Hyderabad India, May 2014.

[42] T. Su, K. Wu, W. Miao et al., "A survey on data-flow testing," *ACM Computing Surveys*, vol. 50, no. 1, pp. 1–35, 2018.

[43] A. Prajapati and J. K. Chhabra, "Information-theoretic remodularization of object-oriented software systems," *Information Systems Frontiers*, vol. 22, no. 4, pp. 863–880, 2019.

[44] A. Rathee and J. K. Chhabra, "A multi-objective search based approach to identify reusable software components," *Journal of Computer Languages*, vol. 52, pp. 26–43, 2019.

[45] M. Paixao, M. Harman, and Y. Zhang, "Multi-objective module clustering for kate," in *Search-Based Software Engineering*, pp. 282–288, Springer, New York, NY, USA, 2015.

[46] L. Pagliari, R. Mirandola, and C. Trubiani, "Engineering cyber-physical systems through performance-based modelling and analysis: a case study experience report," *Journal of Software: Evolution and Process*, vol. 32, no. 1, p. e2179, 2020.

[47] A. Corazza, S. Di Martino, V. Maggio, and G. Scanniello, "Weighing lexical information for software clustering in the context of architecture recovery," *Empirical Software Engineering*, vol. 21, no. 1, pp. 72–103, 2016.

[48] R. Naseem, M. M. Deris, O. Maqbool, and S. Shahzad, "Euclidean space based hierarchical clusterers combinations: an application to software clustering," *Cluster Computing*, vol. 22, no. S3, pp. 7287–7311, 2019.

[49] A. Shahbazian, Y. K. Lee, D. Le, Y. Brun, and N. Medvidovic, "Recovering Architectural Design Decisions," in *Proceedings of the 2018 IEEE International Conference on Software Architecture (ICSA)*, pp. 95–9509, IEEE, Seattle, WA, USA, May 2018.

[50] G. Wang and Q. Song, "Automatic clustering via outward statistical testing on density metrics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 8, pp. 1971–1985, 2016.

[51] M. T. Rahman, P. C. Rigby, and E. Shihab, "The modular and feature toggle architectures of google chrome," *Empirical Software Engineering*, vol. 24, no. 2, pp. 826–853, 2019.