Software Module Clustering Using Grid-Based Many-Objective Particle Swarm Optimization

Amarjeet Prajapati (amarjeetnitkkr@gmail.com)
Jaypee Institute of Information Technology

Research Article

Keywords: Many-objective optimization, grid-based criteria, software clustering, two-archive optimization

Posted Date: June 25th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-407806/v1

Software Module Clustering using Grid-based Many-objective Particle Swarm Optimization

Amarjeet Prajapati
Department of CSE& IT
Jaypee Institute of Information Technology, NOIDA, India
amarjeetnitkkr@gmail.com

Abstract The poor performance of traditional multi-objective optimization algorithms over the many-objective optimization problems has led to the development of a variety of many-objective optimization algorithms. Recently, several many-objective optimization algorithms have been proposed to address the different class of many-objective optimization problems. Most of the existing many-objective optimization algorithms were designed from the perspective of synthetic many-objective optimization problems. Despite the tremendous work made in the development of the many-objective optimization algorithms for solving the synthetic many-objective optimization problems, still real-world many-objective optimization problems gained little attention. In this work, we propose a grid-based many-objective particle swarm optimization (GrMaPSO) for the many-objective software optimization problem. In this contribution, the grid-based selection strategies along with other supportive strategies such as two-archive storing and crowding distance have been exploited in the framework of particle swarm optimization. The performance of the proposed approach is evaluated and compared to three existing approaches over five problem instances. The results demonstrate that the proposed approach is more effective and has significant advantages over existing many-objective approaches designed for the software module clustering problems.

Keywords-Many-objective optimization, grid-based criteria, software clustering, two-archive optimization

1. Introduction

Many-objective software module clustering problem (MaSMCP) is a special case of software module clustering problem (SMCP) that involves optimizing four or more objective functions simultaneously (Mkaouer et al. 2015). The SMCPs containing single objective is commonly referred as single-objective SMCPs (SoSMCPs) (Mitchell and Mancoridis, 2006) and SMCPs containing more than one objective is commonly referred as multi-objective SMCPs (MoSMCPs) (Praditwong et al. 2011). In real-world software engineering, the SMCPs associated with the different software engineering purposes such as software remodularization, architecture recovery, refactoring, restructuring, etc often require optimization of four or more objective functions simultaneously. Therefore, MaSMCPs is an important and crucial form of the optimization problem of the software engineering field that must be given special attention.

To address the different aspects of MaSMCPs, several many-objective software module clustering approaches (MaSMCAs) have been proposed (e.g., Praditwong et al. 2011, Kumari and Srinivas, 2016; Amarjeet et al. 2017;Amarjeet et al. 2018). Most of the existing MaSMCAs is based on the traditional multi-objective optimization approach (MoOA) framework such as NSGA-II (Deb et al. 2002) and multi-objective harmony search (Praditwong et al. 2011). However, the traditional MoOA frameworks are suited well only for the multi-objective optimization problems (MoOPs) consisting limited number of objectives especially less than four objectives. These approaches generally face various difficulties with optimization problems having more than three objectives (Deb et al. 2014). In general, the MoOPs consisting of more than three objectives are commonly referred to as many-objective optimization problems (MaOPs) (Gong et al. 2016; Zhou et al. 2018).

In the literature of metaheuristic search optimization, there are several strategies and concepts are available which are widely exploited to design the many-objective optimization algorithm (MaOA) for the different science and engineering MaOPs. These strategies and concepts are generally categorized as follows: 1) decomposition-based strategy, e.g., the multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li 2007) and

cellular multi-objective genetic algorithms (cMOGA) (Murata et al. 2001); 2) density estimation based or modified Pareto dominance, e.g., shift-density-based evolutionary algorithm (SDEA) (Li et al. 2014), -dominance-based algorithm (Deb et al. 2005), and grid-based evolutionary algorithm (GrEA) (Yang et al. 2013); 3) preferences-based algorithms, e.g., preference inspired co-evolutionary algorithms (PICEA-g) (Wang et al. 2013); 4) Performance indicator based, e.g., approximated hyper volume-based evolutionary algorithm (HypE) (Bader and Zitzler, 2011), indicator-based evolutionary algorithm (IBEA) (Zitzler and Kunzli, 2004). Apart from these categories, there are some more promising approaches, e.g., improved two-archived evolutionary algorithms (Wang et al. 2015), knee point-driven evolutionary algorithm (Zhang et al. 2015), etc.

The aforementioned many-objective optimization strategies have been widely adopted to design the MaOAs for the different synthetic as well as real-world science and engineering MaOPs. For example, industrial scheduling problems (Sulflow et al. 2007), molecular design (Kruisselbrink et al. 2009), control system design (Herrero et al. 2009), Brain-computer interfacing (Pal and Bandyopadhyay, 2016), improve existing package design (Prajapati and Chhabra, 2019), Scientific workflow scheduling in cloud computing (Saeedi et al. 2020) etc. The current development in the domain of MaOA has provided a variety of alternatives to address different aspects of MaOPs and many of them have offered bright prospects to this direction.

Although research in the domain of MaOAs has achieved great success in addressing the different aspects of synthetic MaOPs, their effective potential exploitation, customization and application on real-world MaOPs still needed more study. Formulation of the complex real-world problem as a MaOP and designing appropriate many-objective metaheuristic optimizer is promising and ongoing research. The MaSMCP is an inherent MaOP and their solution has a wide range of applications in the software engineering field. Even after huge development in the designing of many-objective metaheuristic approaches, the MaSMCPs gained little attention in the search-based software engineering (SBSE) community. To address the different aspects of MaSMCPs, some approaches have been proposed in the literature (Praditwong et al. 2011; Mkaouer et al. 2015; Amarjeet et al. 2018). The existing approaches have exploited various existing many-objective optimization strategies and have been applied successfully to address the various forms of MaSMCPs, still, there are many promising many-objective optimization strategies did not gain any attention in this direction. The high importance of MaSMCPs in the software engineering motivates us to explore and design of more effective many-objective optimization approaches for the MaSMCPs by exploiting the most effective many-objective optimization framework and strategies.

From the perspective of complex optimization problems such as discrete and multimodal MoOPs, the swarm intelligence (SI) algorithms are an effective and well suitable approach (Yue et al. 2018). The particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), a class of SI algorithm has been widely successfully applied to address the different discrete and multimodal MoOPs (Coello and Lechuga, 2002; Martínez and Coello, 2011; Moubayed et al. 2014; Dai et al. 2015). The supremacy of PSO framework towards the many-objective optimization problem have also been validated in the various studies (e.g., Figueiredo et al. 2016; Hu et al. 2017; Lin et al. 2018). Recently, the study (Li et al. 2020) has been demonstrated that the grid-based ranking strategy (Yang et al. 2013) for selection of personal and global in many-objective PSO framework is an effective strategy for the discrete and multimodal MaOPs. Apart from that, the two-archive-based external storage strategies along with effective updation techniques have also been found as a supportive component for the many-objective metaheuristic algorithms (Wang et al. 2015).

The favourable characteristics of the PSO framework, grid-based selection strategy, and two-archive-based external storage in solving the discrete and multimodal MaOPs, inspired the development of grid-based PSO for the MaSMCPs. The nature of MaSMCPs as the discrete and multimodal many-objective validates the suitability of the

adapted strategies. Overall, in this study, the various existing strategies favourable to the MaSMCPs have also been integrated into the framework of the PSO. The major contributions of this study can be summarized as follows:

- An effective many-objective metaheuristic search optimizer named as grid-based many-objective particle swarm optimization (GrMaPSO) has been proposed for the MaSMCP.
- To balance the convergence and diversity in GrMaPSO, the procedure of determining the personal best position and global best selection strategy has been redefined based on the grid-based fitness computation (i.e., grid ranking, grid crowding distance, and grid coordinate point distance).
- To balance the exploitation and exploration, definitions of the operators used for computing the new velocity and position of particles in the swarm have been redefined according to the suitability of the characteristics of MaSMCPs.
- To produce a sample of best non-dominated solution (i.e., a well-distributed approximation of Pareto front) that can be a good representative of all non-dominated solutions of the search space, a two-archive external non-dominated storing strategy has been used.

The rest part of this work is designed as follows. Section 2 presents related works corresponding to the search-based software module clustering. Section 3 explains the basic background of the concepts and strategies used in the proposed approach. Section 4 describes the proposed GrMaPSO approach. Section 5 presents the experimental setup used in the GrMaPSO approach. Section 6 presents results and analysis. Section 7 discusses the possible threats and their mitigation. Section 8 concludes the paper with a discussion of future works.

2. Related works

Over the last two decades, many researchers and academicians of the software engineering community, has exploited the benefits of various efficient search-based optimization algorithms like metaheuristic algorithms to support the automation of the different large and complex SMCPs. Based on the different categories of SMCPs, we divide the related works into: single-objective software module clustering approaches (SoSMCAs), Multi-objective software module clustering approaches (MoSMCAs), and Many-objective software module clustering approaches (MaSMCAs).

SoSMCAs: In the direction of SoSMCP, the work (Mancoridis et al. 1998) is considered as the base study where the authors exploited the benefits of search-based optimization algorithm to automate the SMC for source code architecture extraction. To this contribution, they considered the SMCP as a graph partitioning problem where high-level module dependency graph (MDG) is treated as an abstract description of the source code structure. To guide the optimization process modularization quality (MQ) is designed and incorporated into the hill-climbing and genetic algorithm as a fitness function. Motivated by the (Mancoridis et al. 1998), many other researchers and academicians, presented several other aspects of SoSMCAs. The works (Doval et al. 1999) presented a more enhanced version of the SoSMCAs where they customized genetic algorithm with more effective strategies.

To understand the influence of different operators and parameters on the performance of SoSMCA (Mancoridis et al. 1998; Doval et al. 1999), the authors (Mitchell and Mancoridis, 2002) conducted an empirical study. The works (Mahdavi et al. 2003) used the same optimization model (Mancoridis et al. 1998; Doval et al. 1999) for the formulation of SMCP, but they used the multiple hill-climbing algorithms as a metaheuristic search optimizer. The robust fitness function in the presence of software uncertainty is highly required in the search-based optimization. To evaluate the robustness of different existing fitness functions (i.e., MQ and EVM) designed for the SoSMCPs, authors (Harman et al. 2005) conducted an empirical study. To increase the automation and usefulness of the different SoSMCAs, authors (Mitchell and Mancoridis, 2006) developed a framework named as Bunch. In the Bunch framework, the genetic algorithm, hill-climbing, and simulated annealing are integrated as the optimization algorithm.

Apart from designing the robust fitness function, the appropriate encoding also plays an important role in the SoSMCP. The works (Praditwong 2011) introduced a Grouping Genetic Algorithm (GGA) for the SoSMCP. The GGA used a group encoding method to represent the candidate solutions for the SMCP. In many SoSMCAs, direct link-based fitness has been widely used. In contrast to the direct link-based fitness, authors (Huang et al. 2016) utilized a similarity-based fitness evaluation approach to measure the quality of candidate solutions. Recently, some researchers contributed to SoSMCAs by designing and tailoring novel metaheuristic search algorithms. For example, authors (Prajapati and Chhabra, 2017) designed a harmony search-based SoSMCA to address the SoSMCPs and the authors (Pourasghar et al. 2020) designed the graph-based clustering approach to address the SoSMCPs.

MoSMCAs:The development of multi-objective optimization concepts for the SMCPs made tremendous growth in the literature of SMC. The authors (Praditwong et al. 2011) were the first who developed the concepts of multi-objective formulation for solving the SMCP as a MoOP. They developed a different set of multi-objective formulations for the SMCPs and suggested the use of MoOA. To validate the effectiveness of inclusion and exclusion some objectives of the multi-objective formulations (Praditwong et al. 2011), author (Barros 2012) conducted an empirical study. To evaluate the sensitivity of objectives of the multi-objective formulations (Praditwong et al. 2011), authors (Prajapati and Chhabra, 2014) performed a sensitivity analysis over the different SMCPs.

Since the incarnation of multi-objective optimization concepts for the SMCPs, a huge growth has been observed in the designing and customization of multi-objective search methods among the search-based software research community. The works (Kumari and Srinivas, 2016) proposed a hyperheuristic-based MoOA where the authors exploited the hyperheuristic concepts in designing the MoOA for the MoSMCPs. The authors (Prajapati and Chhabra, 2017) exploited the multi-objective optimization framework of the NSGA-II algorithm and customized it for the MoSMCPs. Apart from the customization of effective MoSMCA, they also presented an effective method to compute the weighted class connection strength for the coupling and cohesion.

Recently, many researchers and academician presented several improvements in the formulation of multi-objective optimization as well as designing of the multi-objective optimization algorithms for the MoSMCPs. The works (Jalali, et al. 2019) used the structural and non-structural source code feature to model the multi-objective formulation of the SMCP. They used the multi-objective metaheuristic optimization algorithms to optimize the defined objective functions. The works (Prajapati et al. 2020) used the different aspects of source code information to design the different competing objective functions for the multi-objective formulation of the software module restructuring problem. They used the NSGA-II, multi-objective evolutionary algorithm to optimize the defined objective functions simultaneously. The works (Prajapati and Geem, 2020) proposed the harmony search-based multi-objective software module clustering for software architecture reconstruction. The works (Prajapati and Kumar, 2020) tailored the particle swarm optimization algorithm to address the MoSMCP for software remodularization.

MaSMCAs:Many of the search-based software module clustering approaches are designed by considering only a few aspects of software quality criteria as objective functions. Moreover, such approaches generally treated SMCPs as a MoOP. However, in the real world, the SMCPs employed for different software engineering purposes such as software remodularization and architecture recovery mostly involve a large number of objectives. Therefore, such MoSMCAs cannot be effective for the MaSMCPs. The authors (Mkaouer et al. 2015) were believed to be the first who considered the SMCP as a MaSMCP and solved it by applying the NSGA-III, a many-objective evolutionary algorithm, for software remodularization. Later, some researchers have also contributed to this direction by exploiting the existing framework and strategies. Recently, authors (Prajapati and Chhabra, 2018) used the concept of fuzzy-Pareto dominance to design the artificial bee colony based many-objective optimization approach for the MaSMCP. Further, the same

authors (Prajapati and Chhabra, 2019) exploited the potential of harmony search algorithm along with the several supportive strategies to design the many-objective optimization approach for the MaSMCP.

In the past one decade, a large number of many-objective optimization have been developed to solve the different large and complex MaOPs (Bader and Zitzler, 2011; Yang et al. 2013; Wang et al. 2015; Xiang et al. 2017; Gong et al. 2018). Last few years, the many-objective optimization has become a hot topic in the field of metaheuristic search optimization (e.g., Liu et al. 2019; Sun et al. 2019; Liu et al. 2020; Chen et al. 2020; Pan et al. 2020). Even after, huge development in the designing of the variety of many-objective optimizers, their applications in the real-world MaOP such as MaSMCPs gained little attention. Apart from that most of the existing many-objective optimizers are designed by keeping the characteristics of the synthetic many-objective optimization problems. Therefore, these approaches can work well with the synthetic problem instances and may not work well with real-world unrealistic optimization problems.

Even though there is a huge opportunity for the application of many-objective optimization concepts in the field of software engineering, it could not gain sufficient attention of researchers and academicians of these community. There are only a few works carried out in this direction covering some particular aspect of many-objective software engineering problems (e.g., Mkaouer et al. 2015; Prajapati and Chhabra, 2019). Therefore, more study is required to explore the potential applications of different existing MaOAs in solving the various aspects of many-objective software engineering problems. However, the effective use of different strategies developed in support of many-objective optimization in designing of the many-objective software engineering approach is a challenging task. To increase the applicability of existing optimization model and strategies of many-objective metaheuristic algorithms in the field of software engineering, we propose a many-objective optimization framework by exploiting the potential of particle swarm optimization framework, grid-based selection approach, and two-archive external archive storage.

3. Basic backgrounds

This section provides basic descriptions of the problem formulation and the framework and strategies exploited to design the proposed approach.

3.1. Problem formulation

Similar to the other many-objective optimization problems (MaOPs), the many-objective software module clustering problem (MaSMCP) involves optimizing four or more objective functions simultaneously. The MaSMCP can be described similarly as the other common MaOPs are defined. The mathematical description of the MaOP is as follows:

$$\begin{cases} min\ Q(d) = [q_1(d), q_2(d), \ldots, q_M(d)]^T, & M > 3 \\ r_j(d) \geq 0 & j = 1, \ldots P; \\ x_k(d) = 0 & k = 1, \ldots Q; \\ d_i^{Lower} \leq d_i \leq d_i^{Upper} & i = 1, \ldots, n \end{cases} \quad (1)$$

In the context of the MaSMCP, the objective functions $q_1(d), q_2(d), \ldots, q_M(d)$ can be viewed as the clustering quality criteria and the terms $r_j(d)$ and $x_k(d)$ can be denoted with clustering inequality and equality constraints, respectively. The d_i^{Lower} and d_i^{Upper} are the lower and upper bound of the ith decision variable.

To define the decision variables for the MaSMCPs, the problem can be encoded into the integer-based representation. The description of integer-based representation for an object-oriented software system (especially Java-based system) is presented in Fig. 1.

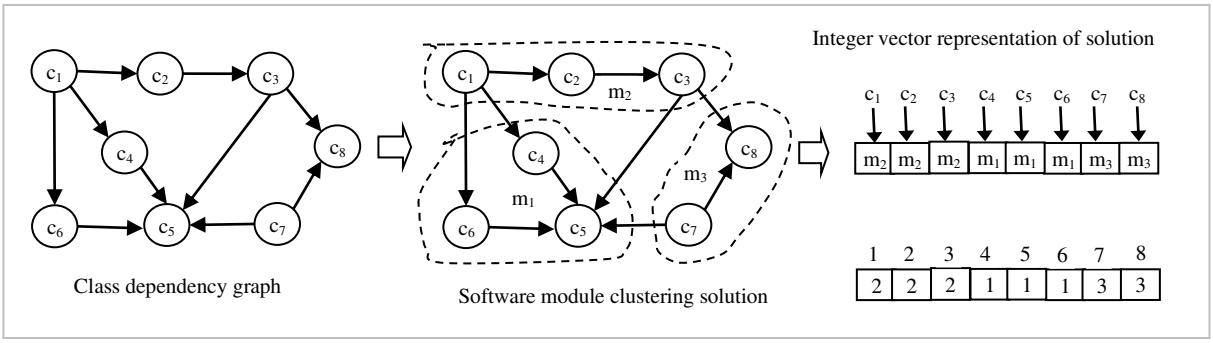

Fig. 1 Software module clustering solution encoding

In the demonstration of the integer-based software module clustering, we consider an artificial object-oriented software system consisting of eight source code classes, i.e., c_1, c_2, c_3, c_4, c_5, c_6, c_7, and c_8. These eight source code classes are connected with some relationships. Now consider we generate a module clustering solution by randomly partitioning these eight classes into three cluster/package, i.e., m_1,m_2, and m_3. Consider in this random software module clustering solution the set of classes {c_4, c_5, c_6} are grouped under cluster {m_1}, the set of classes {c_7, c_8} are grouped under cluster {m_3}, and classes {c_1, c_2, c_3} are grouped under cluster {m_2}. Now if we map the classes, i.e., c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8 with the vector indices 1, 2, 3, 4, 5, 6, 7, 8, and the clusters m_1,m_2, m_3 with the index's values 1, 2, 3, then we can generate a vector with particular values. The values of the under the indices 1, 2, 3, 4, 5, 6, 7, 8, corresponding to the random software module clustering will be 2, 2, 2, 1, 1, 1, 3, 3.

For the MaSMCPs, the previous researchers (e.g., Praditwong et al. 2011; Amarjeet et al.2018) have used the various software quality criteria as the objective functions. In our proposed approach, we consider the following two objective models for the MaSMCP.

- *Extended-Maximizing Cluster Approach (E-MCA):* In the E-MCA objectives model following module clustering criteria are included: 1) number of isolated clusters (minimize), 2) number of clusters (maximizing), 3) sum of inter-edges of all clusters (minimizing), 4) the sum of intra-edges of all clusters (maximizing), 5), modularization quality (MQ) (maximizing), 6) average shortest path length between a source and all other reachable clusters (minimize), 7) cluster cyclic dependencies (minimize).
- *Extended Equal-size Cluster Approach (E-ECA):* In the E-ECA objectives model following module clustering criteria are included: 1) difference between the maximum and minimum number of modules in a cluster (minimizing), 2) number of clusters (maximizing), 3) sum of inter-edges of all clusters (minimizing), 4) sum of intra-edges of all clusters (maximizing), 5), modularization quality (MQ) (maximizing), 6) average shortest path length between a source and all other reachable clusters (minimize), 7) cluster cyclic dependencies (minimize).

The software quality criteria used in the above two objective models have been widely used in the literature of multi-objective software module clustering. Therefore, we have also used these objective models in the many-objective formulation of the software module clustering problem.

3.2. Particle swarm optimization

The particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) is an effective search-based metaheuristic optimizer. The working model of the PSO is inspired by the behaviour of the bird flocking and fish schooling. The PSO framework is highly adaptable and has been customized by the researchers to address the different types of continuous and non-continuous complex science and engineering problems. The individual bird or fish of a swarm is denoted as a moving particle in the spaces. The moving particle in the space is attributed with the velocity and position. To achieve goal or to reach a destination, the particles use their personal experience as well as the swarm experience. In this

activity, every particle continuously changes their velocity. The standard rule of velocity and position updation is as follows:

$$v_i^{new} = \omega v_i + c_1 r_1 (pbest_i - x_i) + c_2 r_2 (gbest - x_i) \quad (2)$$

$$x_i^{new} = x_i + v_i^{new} \quad (3)$$

where ωrepresent the inertia applied to restrain the current speed of the particle. The term $v_i$and $x_i$represent the velocity and position of particle i. The symbol $pbest_i$is the personal best position of ith particle and $gbest$is the global best of the swarm. The constant $c_1 and c_2$ are the learning factors of the personal experience component and social experience components respectively. By changing their velocity and position the particles exploits and explore the search space and finally reaches their expected points/position, i.e., optimal solution of the optimization problem.

3.3. External archives

In the designing of multi/many-objective optimization framework, most of the approaches exploit the potential of the external archive to store the non-dominated solution and guiding the optimization process (e.g., Praditwong and Yao, 2006; Wang et al. 2015). The size of the archives, the number of archives, and the archive management scheme varies from approach to approach as well as according to the problem context. The effective utilization of the potential of the external archive can help the optimization approach to approximate the Pareto front effectively. Many approaches use the fixed-size single external archive to store the non-dominated solutions found in every generation of the optimization algorithm. However, the use of variable size multiple external archives to store the non-dominated solutions and guiding the optimization process have been found more effective in the designing of multi/many-objective optimization framework.

The works (Praditwong and Yao, 2006; Wang et al. 2015) proposed the concept of variable size, two external archives in the designing of multi/many-objective optimization framework. More specifically, they used the two external archives namely convergence archive (CA) and divergence archive (DA). In both archives, non-dominated solutions collected from every generation of many-objective optimizers and stored based on their updating rules. The CA archive helps in achieving better approximation of the Pareto front whereas DA archive helps in distributing the optimal points evenly in the Pareto front. Inspired by the effectiveness of the variable size two external archives, we exploit the potential of these concepts to design our proposed many-objective software module clustering approach.

3.4. Grid-based fitness evaluation

The main goal of every multi-objective search optimizer is to produce a good approximation of the Pareto front. A good approximation of the Pareto front is commonly attributed as a set of solutions closed to the true Pareto front having evenly distributed optimal points. To produce a good approximation of the Pareto front, the multi-objective search optimizers employed various strategies in the optimization process to maintain the diversity and convergence among the candidate solutions. The fitness evaluation of the candidate solutions and its application in mating or environment selection highly affects diversity and convergence of the final results. In the context of the MaOP, the multi-objective metaheuristic search optimizers based on traditional fitness evaluation and selection method generally fail to generate a well-distributed and close approximation of the Pareto front.

A variety of fitness evaluation strategies have been designed for the selection task in the context of many-objective optimization. However, most of them either promote the diversity or convergence not both simultaneously. In such cases, to balance diversity and convergence become difficult. The grid-based fitness evaluation method (Yang et al. 2013) incorporates both convergence and diversity information in determining the candidate solution. Hence, the grid-based method can be a good alternative for the selection of the candidate solution in many-objective optimization.

However, in the literature of many-objective optimization, it gained little attention. In this article, we adapt the grid-based selection strategy suggested in the literature (Yang et al. 2013) for the fitness evaluation. To assign the fitness of all individuals, there are three grid-based criteria namely grid ranking (GR), grid crowding distance (GCD), and grid coordinate point distance (GCPD) is used. The definition of GR, GCD, and GCPD are based on the concepts of grid-dominance and grid-difference. The concept of grid-dominance and grid-difference are defined in terms of a grid frame. The definitions of all these concepts are described as follows:

*Grid setting***-** In grid-based optimization techniques, the grid is viewed as a frame which is used to determine the location of candidate solutions of a population or swarm (in case of PSO) in their objective space. The size of the grid depends on the current candidate solutions of the particular swarm; hence, it varies from swarm to swarm. The grid lower boundary (LB_m) and upper boundary (UB_m) of m^{th} objective of candidate solutions for a swarm S is determined as follows:

$$LB_m = min_m(S) - (max_m(S) - min_m(S))/(2 \times div) \tag{4}$$

$$UB_m = max_m(S) + (max_m(S) - min_m(S))/(2 \times div) \tag{5}$$

where $min_m(S)$ and $max_m(S)$ are the minimum value and maximum value of m^{th} objective among the candidate solutions of swarm S, respectively. The *div* represents the number of partitions created in objective space corresponding to m^{th} objective. Similarly, the lower and upper bound of each objective with their division can be created. The width d_m of each division regarding the m^{th} objective can be computed as follows:

$$d_m = (UB_m - LB_m)/div. \tag{6}$$

Grid location of an individual: After determining the lower boundary, upper boundary, grid division with their width, now we can easily determine the grid location point of a candidate solution 's' in swarm. The grid location or grid coordinate of a candidate solution's' corresponding to the m^{th} objective is computed as follows:

$$GL_m(s) = Floor[(f_m(s) - LB_m)/d_m] \tag{7}$$

where Floor [.] works as the floor function, $f_m(s)$ denotes the actual objective value corresponding to m^{th} objective function.

Grid-dominance: Grid dominance is used to determine the relationship between the two candidate solutions in objective space. It is similar to the Pareto dominance relation but have more relaxation. It is defined in terms of grid location of the candidate solutions. Let p, q ∈S, the solution p grid-dominates to solution i.e., $p \prec_{grid} q$ if and only if

$$\forall i \in (1,2,\ldots,M): GL_i(p) \leq GL_i(q) \wedge$$
$$\exists j \in (1,2,\ldots,M): GL_j(p) < GL_j(q) \tag{8}$$

Where M is the number of objective functions and the grid is constructed for the individual candidate of swarm S.

Grid difference: Grid-difference is used to determine the difference between the two candidate solutions based on their grid coordinates. Let p, q ∈S, the grid-difference between solution p and q is defined as follows:

$$GD(p,q) = \sum_{m=1}^{M} |GL_m(p) - GL_m(q)|. \tag{9}$$

Grid ranking: The grid-ranking (GR) is used to determine the rank of individual of swarm population. The GR is defined as the aggregation of individual's grid coordinate in each objective.

$$GR(s) = \sum_{m=1}^{M} GL_m(s) \tag{10}$$

where $GL_m(s)$ represents the grid location of candidate solution s in m^{th} objective, M denotes the number of objective functions.

Grid crowding distance (GCD): The grid crowding distance is used to estimate the density of a candidate solution with respect to the distribution of neighbors. The GCD of an individual p is defined as follows:

$$GCD(p) = \sum_{q \in N(p)} (M - GD(p,q)) \tag{11}$$

Where *M* denotes the number of objectives $N(p)$ represents the set of neighbors of p.

Grid coordinate point distance (GCPD): The GCPD computes the normalized Euclidean distance between a candidate solution of their objective space and utopia point in its hyperbox. The GCPD of an individual p of swarm S is defined as follows:

$$GCPD(p) = \sqrt{\sum_{m=1}^{M}((f_m(p) - (LB_m + GL_m(p) \times d_m))/d_m)^2} \quad (12)$$

Where $f_m(p)$ and $GL_m(p)$ represent the actual objective value of candidate solution p and grid coordinate, respectively, in the m[th] objective. d_m and LB_m denote the width of hyperbox and lower boundary of grid, respectively, for the m[th] objective.

4. Framework of the proposed work

The framework of the proposed work is primarily divided into two major components: 1) encoding of the problem, and 2) application of GrMaPSO. The first part includes the extraction of software entities and their dependencies, the formation of the MDG, the encoding of the problem as a candidate solution, and the representation of objective vectors. The second part consists of the design of various parts of GrMaPSO. The abstract description of the proposed approach is depicted in Fig.2 and the detailed working descriptions of each component are given in subsequent sub-sections.

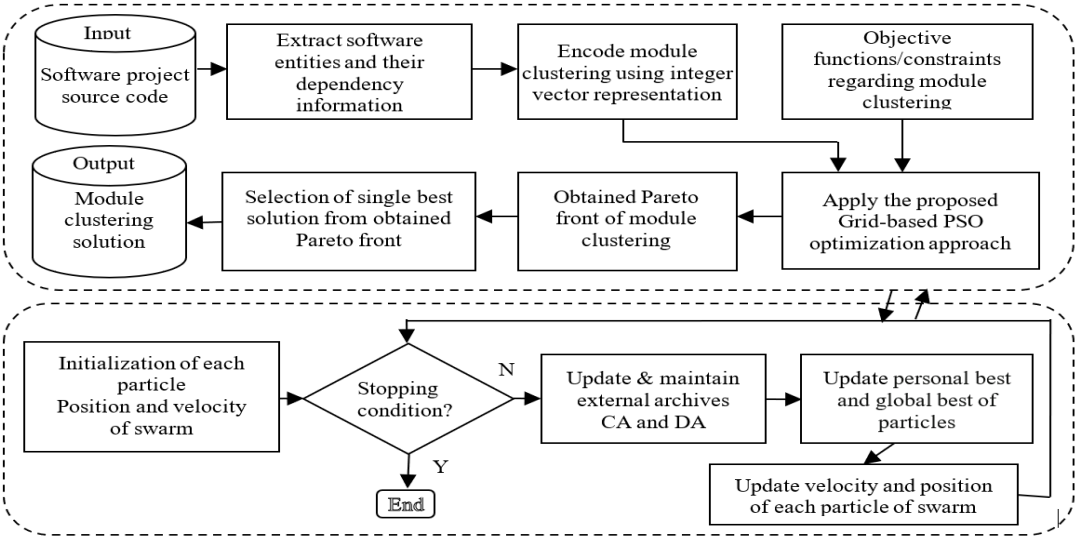

Fig. 1 framework of the proposed work

4.1. Construction of problem optimization model

In the metaheuristic optimization, the problem to be optimized must be represented or encoded into a suitable format so that the operators of the algorithm can be applied effectively. To develop an optimization model for a problem as an input for a metaheuristic algorithm, the different essential components of the optimization problem such as the definition of decision variables, range of decision variables, constraints, and objective functions must be defined.

The proposed approach is specially focusing on clustering of source code classes of the object-oriented system into packages; therefore, we define the classes as entities and method calls, inheritance, etc. between classes as relationships. To extract this information from the source code of object-oriented software systems especially developed in Java programming language, we use the PF-CDA and Structure 101 tool. After extracting the classes and their relationships from the source code, a class dependency graph (CDG) is formed. In the CDG, the classes are denoted with the nodes and class relationships with the graph edges. Consequently, the overall software system to be clustered is transformed into higher-level abstraction i.e., the class dependency graph. The encoding of the problem as a candidate solution and the objective model has been discussed in Section 3.1.

4.2. GrMaPSO

There are many variants of PSO algorithm designed for the different types of science and engineering optimization problems (e.g., single, multi, or many-objective, continuous, discrete optimization, etc.). It is well known that the particular metaheuristic framework and its associated strategies designed for a specific class of optimization problem may not work well with the other types of optimization problems, because, each variant has some merits and limitations when they deal with a particular type of optimization problem. But it is possible to design an effective approach by customizing the existing framework of metaheuristic optimizers with suitable strategies for a particular optimization problem. The MaSMCP is a special kind of discrete combinatorial optimization problem which needs special care while designing the metaheuristic search optimizer.

A new GrMaPSO for the MaSMCP is proposed by exploiting the existing framework of PSO and the various suitable selection and removal strategies. The proposed GrMaPSO has two main features: 1) grid-based selection strategy for the personal best position and global best position, which helps in guiding the GrMaPSO towards the Pareto front, 2) external archives CA and DA with effective updating and pruning strategies, which helps the GrMaPSO to produce a well diverse Pareto front. The proposed GrMaPSO differ from the existing many-objective software module clustering approaches on the determination of personal best and global best position as well as archiving strategy. In this approach, every aspect of SMCP corresponding to many-objective optimization have also been considered while exploiting and redesigning the existing many-objective strategies.

The general framework of the proposed approach is similar to most of the PSO-based multi-objective optimization algorithms, i.e., initialization of swarm, updation and maintenance of the external archive, updation of personal best and global best position, updation of current velocity and position of each particle, and generation of the swarm for a new generation. So, in GrMaPSO, first, the position and velocity of the particles are randomly generated to form an initial swarm. Then, the objective of each particle of the swarm is computed and the external archives CA and DA are updated and pruned if required. Next, the personal best and global best position is updated according adapted strategies. Finally, the velocity and position of the particles of the current swarm are updated for the next iteration. The major steps involved in the proposed GrMaPSO are provided in Algorithm 1.

Algorithm 1: Procedure of proposed grid-based PSO

Input: Swarm size *N*, Number of iterations *itr, Convergence Archive CA, Divergence Archive DA*

Output: External archive CA and DA.

1. Set CA=∅, DA=∅ // In beginning the external archives are empty
2. for i=1 to N do // N is the size of the swarm
3. for j=1 to D do // D is the dimension of the candidate solution vector
4. p_i^j = RandInt (1 to D) // randomly initializes the position of each particle of swarm.
5. v_i^j = RandInt(0 to 1) // randomly initializes the velocity of each particle of swarm.
6. end for
7. Compute the objectives values for each particle
8. Set $p_i^{best} = p_i$ as the personal best position for particle p_i
9. end for
10. Collect all non-dominated solutions from swarm and update the CA and DA
11. Set g^{best} =non-dominated solutions having best grid rank
12. Set itr=0;
13. while itr< MaxItr
14. for i=1 to N do
15. Update particles position and velocity
16. end For
17. Collect all non-dominated solutions from swarm and update the CA and DA
18. Update Personal best using Algorithm 3
19. Update Global best using Algorithm 4
20. Itr=itr+1
21. end while

Algorithm 2: Updating CA and DA

Parameters: S-Swarm, N-Total size of CA and DA

1. Set S_{nd} the non-dominated set of S.
2. $E_{ar} = CA \cup DA$
3. for i=1 to $|S_{nd}|$
4. if $S_{nd}\,[i]$ is not dominated by any solution of E_{ar}
5. if $S_{nd}\,[i]$ dominates one or more solutions of E_{ar}
6. Delete the dominated solutions of CA and DA, and
7. Move the $S_{nd}\,[i]$ solution into the CA
8. else
9. Move $S_{nd}\,[i]$ solution into the DA
10. end if
11. else
12. Discard the $S_{nd}\,[i]$ solution
13. end if
14. end for
15. if |the number of solutions exceeds the size of CA and DA.
16. Delete the extra solution from DA using crowding distance strategy
17. end if

Initialization of particle's position and velocity: The working of proposed grid-based PSO begins with initialization of swarm, i.e., initialization of each particle's position and their velocity resided in the swarm. The initialization of the particle's position and their velocity highly affects the convergence speed of the solutions towards optimal solutions. It is commonly known fact that the initialization of particle position which is uniformly distributed in the search space,

leads optimization process towards optimal front effectively. In this study, we use the random strategy to initialize the position and velocity of the particles in swarm. To initialize the position vector, the index value of the solution vector is selected randomly from their range values, i.e., between 1 to n (number of software entities). To initialize the velocity vector, the index value is selected randomly either 0 or 1.

Updating and pruning external archives: The best non-dominated solutions generated in every generation are collected from the swarm. In the beginning, if the external archives CA and DA are empty, then simply move all the collected solutions into the DA archives. Otherwise, compare each of the collected solutions with the solutions stored in the external archives CA and DA. If the collected solution dominates the one or more solutions of the CA and DA, then remove the dominated solutions from CA and DA and move the collected solution in the CA archive. If the collected solution non-dominates with all solutions of the CA and DA, then move the collected solution in the DA archive. If the collected solution is dominated by solutions of the CA and DA, simply discard the collected solution. Here the size of the individual CA and DA archives are equal and fixed. The number of candidate solutions in the DA archives can exceed the size of the DA. In this case, we use the crowding distance strategy (Deb et al. 2002) to remove the extra solutions from the DA. The pseudo-code of the updating and maintaining rule of CA and DA archive is given in Algorithm 2.

Particle velocity and position updating rules: Updating the velocity and position of particles for the evolution of swarm is an important activity in PSO-based metaheuristic optimizer. The updating procedure highly depends on the nature and representation of the position and velocity of the corresponding optimization problem. For the SMCP, the updation of velocity and position is defined as follows:

$$v_i^{\text{new}} = \Omega(\omega \times v_i + c_1 r_1 \times (\text{pbest}_i \oplus p_i) + c_2 r_2 \times (\text{gbest} \oplus p_i)) \tag{13}$$

$$p_i^{\text{new}} = p_i \Theta \text{v}_i^{\text{new}} \tag{14}$$

where ω is the inertia weight and constant, c_1 and c_2 are the learning factors. The parameters r_1, and r_2, are random numbers distributed randomly between 0 and 1. The pbest_i is the personal best position of the ith particle and gbest is the global best position of the swarm. The operator $\oplus$ behaves like XOR operator and operates between two position vectors and generate new vector. If the ith index value of the both position vector is same the operator $\oplus$produces value 0 for the ith index of new vector, otherwise, it generates value 1. The operator $\times$behaves same as the multiplication operator. The function $\Omega(.)$is defined over a vector containing the real-values and generate new vector containing either value 0 or 1. If the particular index value of the vector is greater than 1, then it produces value 1 otherwise 0 for new vector. The operator $\Theta$is used between position vector and velocity vector which is defined as follows:

$$\begin{cases} p_i^{new} = p_i^{old} & if\, v_i = 0 \\ p_i^{new} = Random\left(a_i^1, a_i^2, \ldots, a_i^{|CA+DA|}\right) & if\, v_i = 1 \end{cases} \tag{15}$$

Updating personal best position: The personal best position of a particle is an important contributor in determining the new velocity and position of that particle. The personal best position is updated if there is an improvement in the new position. To determine the personal best for the intermediate iteration, a grid environment for the combined current swarm and the current set of personal best solution is created and the grid fitness of combined candidate solution is also computed. Algorithm 3 gives a detailed procedure of updating the strategy of personal best position of any particle in the swarm. If the new position Pareto or grid dominates the current personal best position, the current personal best position is replaced with a new position. If the current personal best position and the newly generated position are non-dominated with each other regarding grid dominance and Pareto dominance relations then we use the GCD metric for selection. In this case, the position having a lower GCD value is selected. If GCD value fails to differentiate the position, the selection is done randomly.

Algorithm 3: Updating personal best of each particles

Requires: p_i: new position, p_i^{best}:current personal best position of particle i.

```
for i=1 to N do                                  // swarm size
    if p_i ≺ p_i^best or p_i ≺_grid p_i^best then
        Replace p_i^best with p_i
    else If p_i ≻ p_i^best or p_i ≻_grid p_i^best then
            Continue without any change
    else if GCD (p_i) < GCD (p_i^best)
        Replace p_i^best with p_i
    else if GCD (p_i) > GCD (p_i^best)
        Continue without any change
    else If random (0,1) < 0.5 then
        Replace p_i^best with p_i
    else
        Continue without any change
    end if
end for
```

Algorithm 4: Updating global best solution

Input: $p^{best} = \{p_1^{best}, p_2^{best}, \ldots, p_N^{best}\}$, CA, DA, G=∅, H=∅, $g_i \in G$

Output: g^{best} (best solution in G)

```
Combine all solutions of CA, DA and p^best into H.
Collect all non-dominated solutions from H to G
g^best ← g_1
for i = 2 to |G| do
    if GR (g_i) < GR (g^best) then
        g^best ← g_i
    else if GR (g_i) = GR (g^best) then
        if GCD (g_i) < GCD (g^best then
            g^best ← g_i
        else if GCD (g_i) = GCD (g^best) then
            if GCPD (g_i) < GCPD (g^best) then
                g^best ← g_i
            end if
        end if
    end if
end for
```

Selection of global best position: In this study, the global best position is selected from the all-current personal best solutions and solutions stored in the CA and DA archives. First, the candidate solutions corresponding to all personal best and of CA and DA are combined into an auxiliary storage. Then all best non-dominated solutions are collected from the auxiliary storage. Next, a grid environment is set for all non-dominated solution and each individual of the collected non-dominated solutions are hierarchically compared according to the GR, GCD, and GCPD fitness criteria. Finally, a single best solution is returned which is considered as the global best position. The detailed procedure of the selection of the global best position is provided in Algorithm 4.

5. Experimental design

This section presents the experimental setup designed for the proposed GrMaPSO. This experimental setup includes a selection of test problems, competitor algorithms, result collecting procedure, and statistical test.

5.1. Test problems

The test problems include a variety of software projects with different size and characteristics. The proposed approach can be applied to any type of software projects, but this study is mainly focussing on the software projects especially developed in the object-oriented programming paradigm. The selected software projects are Java Servlet API, Junit, XML API DOM, JavaCC, DOM4J, JHotDraw, and JFreeChart. The brief information of these software projects are provided in Table 1.

Table 1 Characteristics of selected software projects

Systems	Version	#Classes	#Dependencies
Java Servlet API	2.3	63	131
JUnit	3.81	100	276
XML API DOM	1.0.b2	119	209
JavaCC	1.5	154	722
DOM 4J	1.5.2	195	930
JHotDraw	6.0b1	398	2175
JFreeChart	0.9.21	401	1420

The main reason for selecting these software projects for the evaluation of our proposed approach is that these software projects are highly used in different application. Additionally, the research community of search-based software engineering field have also used these projects to evaluate the similar research methods as test problems.

5.2. Competitor approaches

In the field of search-based software engineering, various metaheuristic search optimizers have been designed and developed by tailoring or customizing the traditional metaheuristic algorithms to solve the different aspects of software

engineering problems. As our proposed approach is targeting to the many-objective optimization aspect of SMCP, so we have selected only those metaheuristic search optimizers of search-based software engineering to compare our approach which are based on many-objective optimization concepts. The brief descriptions of the selected existing approaches are given as follows:

- *Two-archive Pareto optimal genetic algorithm (TA-PGA) (Praditwong et al. 2011):* The TA-PGA is genetic-based software module clustering approach where external archive concept has been exploited in the traditional multi-objective genetic algorithm.
- *NSGA-III-based software remodularization (NBSR) (Mkaouer et al. 2015)*: This approach was designed to address the many-objective software remodularization problem. In this method, the traditional NSGA-III metaheuristic search optimizer is tailored to optimize the software modularization to improve the quality.
- *Fuzzy Pareto-Dominance Driven Artificial Bee Colony (FP-ABC) (Amarjeet et al. 2018):* This approach was specially designed to address the many-objective software clustering problem. In this approach, fuzzy-Pareto dominance selection strategy has been introduced in the traditional artificial bee colony algorithm.

The basic reason for selecting the above existing many-objective search optimizer is that these metaheuristic search optimizers are directly related to our proposed approach and designed to address the many-objective software engineering problems.

5.3. Parameter settings

The metaheuristic search optimizer generally contains many parameters and their configuration values highly influence the final results. Hence, the parameter setting of these algorithms for a specific problem is a challenging task. To determine the most suitable parameter values of a metaheuristic search optimizer corresponding to a specific problem, various tuning methods are used. In our approach, we have used the trial-and-error approach to determine the parameter values. However, for the existing approaches, we have used the same parameter setting values as used by their designers. The parameter setting values of different optimizers are provided in Table 2.

Table 2Metaheuristic search optimizer and parameters values

#	Many-objective metaheuristic algorithms	Parameters	Values
	Proposed	Swarm size	10*N
		|CA+DA|	10*N
		Number of fitness evaluations	2000* N
		Inertia weight ω	0.6
		Constants c_1 and c_2	0.7 and 1.3
		Division	9
1	TA-PGA	Population Size	10*N
		Crossover weight	0.8 to 1.0
		Mutation weight	0.04 *$\log_2$(N)
		Number of fitness evaluations	2000* N
		|CA+DA|	10*N
2	NBSR	Population Size	10*N
		Number of fitness evaluations	2000* N
		Crossover weight	0.8 to 1.0
		Mutation weight	0.04 *$\log_2$(N)
		Reference points	190
3	FP-ABC	Number of food sources (Population size)	10*N
		Number of fitness evaluations	2000* N
		Size of CA (Convergence Archive)	5*N
		Size of DA (Divergence Archive)	5*N
		Number of employed bees	10*N
		Number of onlooker bees	10*N
		Limit	0.05*N
		Selection probability (p_{ps})	0.05

* N represents the number of modules of the studies software systems.

5.4. Collecting Results and statistical tests

The metaheuristic search optimizers are not deterministic. The inclusion of various randomized components in their design prevents them to behave deterministically. Due to stochastic nature, the metaheuristic search optimizers may not generate the same output, if they are executed multiple times on the same problem input. In this situation, it becomes difficult to conclude the output collected from the metaheuristic search optimizers.

To overcome the problem, researchers of the metaheuristic search community generally suggest the use of the statistical test. Another challenge is the many-objective metaheuristic search optimizers do not produce a single best solution but a set of non-dominated solution (i.e., Pareto set). To select a single solution that exhibits the best trade-off to all objective functions is another challenging task. In this study, we use the trade-off worthiness metric (Rachmawati .and Srinivasan, 2009), a more appropriate approach for the selection of the best solution. We run each metaheuristic search optimizers 31 times and using the trade-off worthiness metric and collect the 31 best solutions as a sample for the statistical test. For the statistical test, we used the Mann-Whitney U test with a 95% confidence level (α=0.05).

5.5. Assessment criteria

To measure the quality of the modular design of a software system, coupling and cohesion are two important design quality metrics. Apart from the coupling and cohesion, modularization quality (MQ) is another highly used design quality measurement in software module clustering. Along with the design quality measurement, we also use the criterion to measure the quality of generated Pareto front of the metaheuristic search optimizers. The brief description of these quality metrics is given below:

- *Coupling and Cohesion Assessment Criterion:* The software system with low coupling and high cohesion is considered a better software design. The coupling measures the degree of dependency of software entities between the modules and cohesion measures the degree of dependency of software entities within the modules.
- *MQ Value as Assessment Criterion:* The MQ measures the trade-off between the coupling and cohesion of the software design. The larger value of the MQ promotes the cohesion and penalizes the coupling. Hence, the software system with higher MQ value is considered the better design compared to the software system with lower MQ value.
- *Pareto Optimality as Assessment Criterion:* The metaheuristic search optimizers produces the results in the form of a set of non-dominated solution which is commonly known as obtained Pareto front. The obtained Pareto front having uniform distribution and closer to the true Pareto front is considered as good Pareto front. To measure the diversity and convergence of the obtained Pareto front the inverse generational distance (IGD) is considered as an effective metric.

6. Results and discussion

This section presents the results of the experimental setup designed for the proposed, GrMaPSO approach and existing approaches. The performance of the GrMaPSO is investigated with three existing many-objective software engineering approaches which are namely the FP-ABC, TA-PGA, and NBSR. Comparative studies of GrMaPSO approach are carried out with the existing approaches to examine their performance on the MaSMCPs. The results of coupling, cohesion, MQ, and IGD of each algorithm is collected according to the method discussed in Section 5.4 and systematically analysed with Mann-Whitney U test to validate the supremacy of the proposed approach compared to the existing approaches. The significant difference between the results of algorithms is determined at the 95% confidence level (for coupling, cohesion, and MQ performance metrics) and 99% confidence level (for IGD performance metrics).

The coupling, cohesion, MQ, and IGD results of the proposed approach is compared with existing, FP-ABC, TA-

PGA, and NBSR approaches. The coupling, cohesion, MQ, and IGD results of the proposed approach compared with existing, FP-ABC, TA-PGA, and NBSR approaches presented in each table are described as follows:1) If the existing approach performs significantly worst compared to the proposed approach then symbol "[-]" is attached with the result of the existing approach, 2) If the existing approach performs significantly better compared to the proposed approach then symbol "[+]" is attached with the result of the existing approach, 3) If there is no significant difference between the proposed approach and existing approach then "[≈]" is attached with the results of the existing approach.

6.1. Coupling as assessment criterion

The coupling results obtained through the proposed and existing approacheswith E-MCA and E-ECA on each of the problem instance are presented in Table 3 and Table 4, respectively. If we see the coupling results of each algorithm obtained with the E-MCA formulation presented in Table 3, the proposed approach is performing significantly better in most of the cases compared to the existing approaches. If we compare the coupling results of the proposed approach and the FP-ABC, the proposed approach produces better coupling in six out of seven cases compared to the FP-ABC approach, in which four cases are significantly better.

The proposed approach produces better coupling in seven out of all seven cases compared to the TA-PGA, in which five cases are significantly better. The proposed approach outperforms the NBSR in all seven cases, in which six cases are significantly better. Similarly, if we see the coupling results achieved with E-ECA many-objective software module clustering formulation, the proposed approach is performing significantly better to the existing approaches in most of the cases. Overall, the coupling results presented in Table 3 and 4 validate that the proposed approach is able to generate module clustering solution having better coupling values compared to the existing approaches.

Table 3 Coupling values achieved with E-MCA many-objective software module clustering formulation

Systems	GrMaPSO	FP-ABC	TA-PGA	NBSR
Java Servlet API	22.15(8.45)	23.23(13.95) [≈]	26.13(8.85)[≈]	25.55(4.78)[≈]
JUnit	75.18 (12.34)	77.29(33.10) [≈]	91.34(14.67)[-]	92.55(10.55)[-]
XML API DOM	64.52(12.76)	69.61(43.85) [-]	76.45(11.37)[-]	79.77(20.61)[-]
JavaCC	203.12(23.43)	218.13(46.87) [-]	227.47(33.44)[-]	230.27(26.51)[-]
DOM 4J	243.32(41.38)	241.84(45.25) [≈]	246.34(74.64)[≈]	251.15(47.61)[-]
JHotDraw	767.32(33.11)	791.71(35.11) [-]	813.43(41.47)[-]	815.451(45.44)[-]
JFreeChart	616.46(84.12)	677.51(53.25) [-]	687.31(12.53)[-]	704.32(46.13)[-]

Table 4Coupling values achieved with E-ECA many-objective software module clustering formulation

Systems	GrMaPSO	FP-ABC	TA-PGA	NBSR
Java Servlet API	20.32 (6.23)	21.23(8.45) [≈]	31.50(8.82)[-]	29.57(8.45)[-]
JUnit	61.45(11.67)	69.17(13.34) [-]	73.31(17.39)[-]	80.45(80.45)[-]
XML API DOM	67.23 (5.23)	75.25(7.36) [-]	77.75(14.08)[-]	79.22(13.17)[-]
JavaCC	223.85(34.11)	225.31(46.14) [≈]	197.85(39.16)[+]	234.45(26.15)[-]
DOM 4J	212.98(55.31)	239.45(68.58) [-]	241.75(75.22)[-]	284.41(42.14)[-]
JHotDraw	713.28(71.45)	757.15(73.24) [-]	803.13(84.15)[-]	817.43(73.28)[-]
JFreeChart	661.95(44.21)	664.24(45.45) [≈]	665.09(48.62)[≈]	691.40(42.21)[-]

6.2. Cohesion as assessment criterion

The cohesion results of both proposed and existing approaches evaluated over seven software projects with E-MCA many-objective software module clustering formulation and E-ECA many-objective software module clustering formulation are given in Table 5 and 6, respectively. The results presented in both Table 5 and 6 corresponding to the E-MCA many-objective software module clustering formulation and E-ECA many-objective software module clustering show that the existing approaches, i.e., FB-ABC, TA-PGA, and NBSR are performing significantly worst in most of the cases compared to the proposed approach.

The cohesion results achieved with the E-MCA show that the proposed approach perform significantly better compared to the FB-ABC, TA-PGA, and NBSR in three, five, and five cases, respectively out of seven cases. Now if we see the cohesion results achieved with the E-ECA, it shows that the proposed approach performs significantly better compared to the FB-ABC, TA-PGA, and NBSR in four, four, and five cases, respectively out of seven cases. There is only one

case, i.e., JavaCC in TA-PGA where the proposed approach is demonstrating the worst performance.

Table 5Cohesion values achieved with E-MCA many-objective software module clustering formulation

Systems	GrMaPSO	FP-ABC	TA-PGA	NBSR
Java Servlet API	108.45(8.89)	109.76(10.85) [≈]	104.67(8.75)[≈]	105.31(4.72)[≈]
JUnit	201.12(12.11)	194.71(13.11) [≈]	184.23(14.67)[-]	183.33(11.52)[-]
XML API DOM	144.18(14.23)	141.78(13.52) [≈]	131.34(12.14)[-]	128.64(21.42)[-]
JavaCC	518.39(24.67)	502.95(24.81) [-]	494.03(31.46)[-]	493.25(26.52)[-]
DOM 4J	688.47(83.53)	682.12(84.26) [≈]	682.06(73.67)[≈]	674.56(48.63)[≈]
JHotDraw	1405.68(37.41)	1383.26(35.05) [-]	1361.89(39.15)[-]	1358.52(45.41)[-]
JFreeChart	1485.78(56.34)	1421.42(51.28) [-]	1413.19(31.51)[-]	1397.48(46.10)[-]

Table 6Cohesion values achieved with E-ECA many-objective software module clustering formulation

Systems	GrMaPSO	FP-ABC	TA-PGA	NBSR
Java Servlet API	106.43 (4.87)	104.97(4.81) [≈]	103.09(8.52)[≈]	103.13(7.31)[≈]
JUnit	214.67(12.38)	206.13(13.38) [-]	202.65(11.35)[-]	195.63(18.31)[-]
XML API DOM	141.59(6.38)	133.61(7.21) [-]	132.14(14.15)[-]	130.15(13.52)[-]
JavaCC	498.31(47.29)	496.18(46.13) [≈]	524.42(49.14)[+]	487.43(26.12)[≈]
DOM 4J	718.85(69.35)	691.45(68.28) [-]	683.26(75.23)[-]	645.32(32.05)[-]
JHotDraw	1462.47(78.93)	1417.90(73.84) [-]	1371.90(74.15)[-]	1357.47(73.48)[-]
JFreeChart	1440.86(43.48)	1437.46(45.45) [≈]	1436.97(48.62)[≈]	1414.40(45.70)[-]

6.3. MQ as assessment criterion

The MQ results corresponding to both E-MCA many-objective software module clustering formulation and E-ECA many-objective software module clustering formulation achieved through the proposed and existing approaches are provided in Table 7 and 8, respectively. Similar to the coupling and cohesion results, the proposed approach is also able to perform better compared to the existing approaches in terms of MQ quality metric. The MQ results presented in Table 7 and 8 clearly show that the proposed approach is performing significantly better compared to the existing approaches in most of the cases.

If we compare the MQ values achieved by each of the approaches with E-MCA in pair wise, it can be seen that the proposed approach performs significantly better compared to the FB-ABC, TA-PGA, and NBSR in three, four, and six cases, respectively out of seven cases. Similarly, if we compare the MQ values achieved by each of the approaches with E-ECA in pair wise, it can be seen that the proposed approach performs significantly better compared to the FB-ABC, TA-PGA, and NBSR in four, four, and six cases, respectively out of seven cases. The significantly better MQ values achieved by the proposed approach indicate that the produced module clustering improves the cohesion and decreases coupling.

Table 7MQ values achieved with E-MCA many-objective software module clustering formulation

Systems	GrMaPSO	FP-ABC	TA-PGA	NBSR
Java Servlet API	3.217 (0.132)	3.215(0.153) [≈]	3.045(0.210)[-]	2.923(0.123)[-]
JUnit	6.864 (0.178)	6.128(0.176) [-]	6.153(0.257)[-]	6.289(0.119)[-]
XML API DOM	5.451 (0.356)	5.450(0.387) [≈]	5.398(0.273)[≈]	5.149(0.391)[-]
JavaCC	6.798 (0.123)	6.737(0.124) [≈]	6.918(0.117) [+]	6.459(0.112)[-]
DOM 4J	12.234(0.534)	11.581(0.639) [-]	11.579(0.392)[-]	11.459(0.497)[-]
JHotDraw	12.193 (0.534)	11.989(0.583) [≈]	11.843(0.482)[-]	13.631(0.589)[≈]
JFreeChart	19.341 (0.323)	18.039(0.319) [-]	18.998(0.195)[≈]	17.598(0.421)[-]

Table 8MQ values achieved with E-ECA many-objective software module clustering formulation

Systems	GrMaPSO	FP-ABC	TA-PGA	NBSR
Java Servlet API	3.781 (0.231)	3.659(0.211) [≈]	3.579(0.143)[≈]	3.271(0.134)[-]
JUnit	7.123 (0.143)	6.507(0.128) [-]	6.227(0.219)[-]	6.317(0.181)[-]
XML API DOM	6.238 (0.467)	5.675(0.552) [-]	5.426(0.312)[-]	5.251(0.313)[-]
JavaCC	6.834 (0.187)	6.965(0.182) [≈]	6.752(0.110)[≈]	6.731(0.205)[≈]
DOM 4J	12.543 (0.534)	11.862(0.675) [-]	11.678(0.725)[-]	11.263(0.548)[-]
JHotDraw	12.348 (0.385)	12.401(0.270) [≈]	12.347(0.394)[≈]	12.367(0.652)[≈]
JFreeChart	20.342 (0.305)	18.212(0.252) [-]	18.189(0.221)[-]	18.046(0.136)[-]

6.4. Pareto optimality as assessment criterion

To test the quality of the achieved approximation set achieved through the proposed and existing many-objective optimizers, we use the IGD metric as Pareto optimality as an assessment criterion. In this section, we compare the proposed approach with the existing approaches in terms of how well each of the many-objective algorithms performs

at producing good approximations to the Pareto front. To achieve the well diverse and converge approximations to the Pareto front, we have used exploited the grid-based approach and integrated into the PSO algorithm. Here the grid-based selection strategy is based on both convergence and divergence information. Therefore, the proposed approach is expected to perform better the existing approaches to produce good approximations to the Pareto front.

The IGD values of the proposed and existing software module clustering approaches for both E-MCA many-objective software module clustering formulation and E-ECA many-objective software module clustering formulation are provided in Table 9 and 10, respectively. The statistical results mentioned in the bracket for all existing approaches clearly show that the proposed approach outperforms the existing approached in most of the cases. For example, if we see the IGD results of proposed and FP-ABC approach for E-MCA many-objective software module clustering formulation, the proposed approach performs significantly better in four cases out of seven cases. Similarly, other comparative results presented in Table 9 and 10, show that the proposed approach can achieve significantly better IGD values compared to the rest of the existing approaches.

Table 9 IGD values achieved with E-MCA many-objective software module clustering formulation

Systems	GrMaPSO	FP-ABC	TA-PGA	NBSR
Java Servlet API	5.894×10^{-4}	5.921×10^{-4}[-]	5.972×10^{-4}[-]	5.982×10^{-4}[-]
JUnit	3.812×10^{-4}	3.913×10^{-4}[-]	3.994×10^{-4}[-]	3.929×10^{-4}[-]
XML API DOM	5.329×10^{-3}	5.492×10^{-3}[-]	5.476×10^{-3}[-]	5.649×10^{-3}[-]
JavaCC	5.194×10^{-3}	5.487×10^{-3}[-]	5.192×10^{-3}[≈]	5.356×10^{-3}[-]
DOM 4J	7.973×10^{-3}	7.975×10^{-3}[≈]	7.998×10^{-3}[≈]	7.999×10^{-3}[≈]
JHotDraw	7.782×10^{-3}	7.785×10^{-3}[≈]	7.954×10^{-3}[-]	7.784×10^{-3}[≈]
JFreeChart	5.238×10^{-4}	5.237×10^{-4}[≈]	5.239×10^{-4}[≈]	5.383×10^{-4}[-]

Table 10 IGD values achieved with E-ECA many-objective software module clustering formulation

Systems	GrMaPSO	FP-ABC	TA-PGA	NBSR
Java Servlet API	6.549×10^{-4}	6.612×10^{-4}[-]	7.011×10^{-4}[-]	7.015×10^{-4}[-]
JUnit	4.107×10^{-4}	4.104×10^{-4}[≈]	4.106×10^{-4}[≈]	4.105×10^{-4}[≈]
XML API DOM	6.191×10^{-3}	6.193×10^{-3}[≈]	6.212×10^{-3}[-]	6.231×10^{-3}[-]
JavaCC	5.127×10^{-3}	5.218×10^{-3}[-]	5.271×10^{-3}[-]	5.271×10^{-3}[-]
DOM 4J	8.002×10^{-3}	8.001×10^{-3}[≈]	8.121×10^{-3}[-]	8.116×10^{-3}[-]
JHotDraw	7.679×10^{-3}	7.672×10^{-3}[≈]	7.753×10^{-3}[-]	7.742×10^{-3}[-]
JFreeChart	5.128×10^{-4}	5.534×10^{-4}[-]	5.129×10^{-4}[≈]	5.261×10^{-4}[-]

Overall, the results presented in section 5.1 to 5.4 corresponding to coupling, cohesion, MQ, and IGD performance metrics demonstrates that the proposed many-objective metaheuristic optimizer is able to produce good module clustering solutions and same time outperform the existing approaches. The balanced exploitation and exploration capability strategy and proper fitness evaluation defined in terms of both convergence and diversity information helps the proposed approach to generate such good results.

7. Threats to validity

In this section, various threats that can affect the validity of the obtained results are discussed. There can be several factors that can be responsible in influencing the validity of results. These factors can be widely divided into two main categories: external and internal threats to validity.

In external validity, the capability of generalization of results over the other set of test problems is considered. In software engineering, there are wider range of software projects developed in different languages. Hence, validation of the proposed approach over the different types of software projects is an important point. In our approach, this threat to validity is mitigated by using the abstract description (dependency graph) of the software system as input. It is possible, that a large number of software systems can be mapped to a single dependency graph. To cover a diverse set of dependency graph as input, we have selected the diverse set of open-source software systems.

In internal validity, various experimental treatments are considered that affects the final results of the algorithms. In this study, various quality measures such as coupling, cohesion, MQ, and IGD have been used. The information used in

computing the coupling, cohesion, and MQ is based on the existing works. These quality measures are widely used to evaluate the software quality. The other factor that can affect the validity of the results is the selection of different parameters values. To mitigate this threat, we determined the different parameter's value of the proposed algorithm based on the trial-and-error method as well as the settings of the existing approaches.

8. Conclusion and future works

In this work, we have presented a many-objective optimization approach named as GrMaPSO for the many-objective software module clustering problem. In this contribution, we have exploited the PSO framework along with grid-based ranking and external archive-based solution storing strategies. The grid-based ranking strategy used for the selection of the swarm leaders (i.e., personal best and global best position) effectively and that helped the GrMaPSO to converge towards Pareto front efficiently. Moreover, the concept of two external archives (i.e., convergence-oriented archive and divergence-oriented archive) along with effective updation and pruning strategies have also been incorporated in the GrMaPSO for balancing the diversity and guiding the optimization process.

In the experimental studies, we applied the proposed GrMaPSO over the five MaSMCPs under the E-MCA and E-ECA many-objective formulation. The obtained results are compared with the three-existing many-objective optimization approaches designed for similar many-objective optimization problems. The results indicate that the proposed GrMaPSO outperforms the existing many-objective optimization approaches in terms of the MQ, coupling, cohesion, and IGD quality indicator metrics. Overall, the results demonstrated that the proposed approach is more effective and has significant advantages over existing search-based software module clustering approaches. The future work for this work can be the evaluation of the proposed approach over some other industrial MaSMCPs where a greater number of module clustering criteria are required.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

References

Mark Harman and Bryan F. Jones. 2001. Search-based software engineering. *Inf. Software Technol.* 43, 14 (2001), 833–839.

Harman, M., Afshin Mansouri, S., Zhang, Y., 2012. Search based software engineering: trends, techniques and applications. ACM Comput. Surv. 45 (1), 1–64.

A. Sulflow, N. Drechsler, and R. Drechsler, “Robust multiobjective ¨ optimization in high dimensional spaces,” in Proc. Evol. Multi-Criterion Optimization, 2007, pp. 715–726.

J. W. Kruisselbrink, M. T. Emmerich, T. Back, A. Bender, A. P. Ijzerman, ¨ and E. Horst, “Combining aggregation with Pareto optimization: A case study in evolutionary molecular design,” in Proc. Evol. Multi-Criterion Optimization, 2009, pp. 453–467

J. G. Herrero, A. Berlanga, and J. M. M. Lopez, “Effective evolutionary ´ algorithms for many-specifications attainment: Application to air traffic control tracking filters,” IEEE Trans. Evol. Comput., vol. 13, no. 1, pp. 151–168, Jan. 2009.

M. Pal, S. Bandyopadhyay, Reliability of convergence metric and hypervolume indicator for many-objective optimization, in: Control, Instrumentation, Energy & Communication (CIEC), 2016 2nd International Conference on, IEEE, 2016, pp. 511–515,

Mkaouer, M, Kessentini, M., Shaout, A., Koligheu, P., Bechikh, S, Deb, K, Ouni, A (2015) ‘Many objective software remodularization using NSGA-III, ACM Transaction on software engineering and methodology, Vol. 24 No.3, pp.1-17.

T. Murata, H. Ishibuchi, and M. Gen, “Specification of genetic search directions in cellular multi-objective genetic algorithms,” in *Evolutionary Multi-Criterion Optimization*, E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, Eds. Berlin, Germany: Springer, 2001, pp. 82–95.

Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

M. Li, S. Yang, and X. Liu, “Shift-based density estimation for Paretobased algorithms in many-objective optimization,” *IEEE Trans. Evol.Comput.*, vol. 18, no. 3, pp. 348–365, Jun. 2014.

K. Deb, M. Mohan, and S. Mishra, “Evaluating the epsilon-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions,” *Evol. Comput.*, vol. 13, no. 4, pp. 501–525, 2005.

S. Yang, M. Li, X. Liu, and J. Zheng, “A grid-based evolutionary algorithm for many-objective optimization,” *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 721–736, Oct. 2013.

R. Wang, R. C. Purshouse, and P. J. Fleming, “Preference-inspired coevolutionary algorithms for many-objective optimization,” *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 474–494, Aug. 2013.

J. Bader and E. Zitzler, “HypE: An algorithm for fast hypervolume-based many-objective optimization,” *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.

E. Zitzler and S. Kunzli, “Indicator-based selection in multiobjective search,” in *Parallel Problem Solving From Nature—PPSN VIII* (LNCS 3242), X. Yao *et al.*, Eds. Heidelberg, Germany: Springer, 2004, pp. 832–842.

X. Zhang, Y. Tian, and Y. Jin, “A knee point-driven evolutionary algorithm for many-objective optimization,” *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 761–776, Dec. 2015.

H. Wang, L. Jiao, and X. Yao, “Two_Arch2: An improved two-archive algorithm for many-objective optimization,” *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 524–541, Aug. 2015.

Praditwong K, Harman M, Yao X. Software module clustering as a multi-objective search problem. IEEE Transaction on Software Engineering, 2011, 37(2), pp. 264–282.

Mitchell, B.S., Mancoridis, S. On the automatic modularization of software systems using the bunch tool. IEEE Transactions on Software Engineering, 2006; 32(3), pp. 193–208.

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

Amarjeet Prajapati, Jitender Kumar Chhabra, TA-ABC: Two-Archive Artificial Bee Colony for Multi-objective Software Module Clustering Problem, Journal of Intelligent Systems, (2017), pp. 1-21, DOI: https://doi.org/10.1515/jisys-2016-0253

Amarjeet Prajapati,Jitender Kumar Chhabra, MaDHS: Manyobjective discrete harmony search to improve existing package design. Computational Intelligence. 2019; 35: 98– 123.

Amarjeet prajapati, Jitender Kumar Chhabra, “FP-ABC: Fuzzy Pareto-Dominance Driven Artificial Bee Colony Algorithm for Many Objective Software Clustering”, Computer Languages, Systems & Structures,Volume 51, January 2018, p 1-21.

K. Deb, “Multi-objective optimization,” in *Multi-Objective Optimization Using Evolutionary Algorithms*. Boston, MA, USA: Springer, 2001, pp. 13–46.

Kennedy, J.; Eberhart, R.: Particle swarms optimization. In: Proceedings of 1995 IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)

K. Praditwong and X. Yao, “A new multi-objective evolutionary optimization algorithm: The two-archive algorithm,” in Proc. Int. Conf. Comput. Intell. Security, vol. 1. Guangzhou, China, 2006, pp. 286–291.

H. Wang, L. Jiao and X. Yao, "Two_Arch2: An Improved Two-Archive Algorithm for Many-Objective Optimization," in *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 524-541, Aug. 2015

L. Rachmawati. D. Srinivasan, Multiobjective evolutionary algorithm with controllable focus on the knees of the Pareto front. IEEE Transaction on Evolutionary Computation. 13(4) (2009), pp.810–824.

Abdeen H, Sahraoui H, Shata O, Anquetil N, Ducasse S. Towards automatically improving package structure while respecting original design decisions. *20th Working Conference on Reverse Engineering (WCRE)*, Koblenz, 2013, pp. 212-221

P. Andritsos and V. Tzerpos, "Information-theoretic software clustering," in *IEEE Transactions on Software Engineering*, vol. 31, no. 2, pp. 150-165, Feb. 2005, doi: 10.1109/TSE.2005.25.

Deb K., Pratap A., Agarwal S., Meyarivan T. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. On Evolutionary Computation, 2002, 6 (2): 182-197.

Zitzler E., Laumanns M., Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In:Proc. of the Evolutionary Methods for Design, Optimisation and Control. Athens: International Center for Numerical Methods in Engineering, 2002. 95-100.

Deb K., Jain H. An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part I: solving problems with box constraints, IEEE Trans. Evol. Comput., 2014, 18 (4): 577–601.

Zhou Y, Yi X, Chen Z, et al. A Scalar Projection and Angle-Based Evolutionary Algorithm for Many-Objective Optimization Problems[J]. IEEE Transactions on Cybernetics, 2018, (99):1-12.

Gong YJ, Li JJ, Zhou Y, Li Y, Chung HSH, Shi YH, and Zhang J. Genetic learning particle swarm optimization. IEEE transactions on cybernetics, 2016, 46(10): 2277-2290.

Fang W, Zhang L, Yang S, et al. A Multiobjective Evolutionary Algorithm Based on Coordinate Transformation. IEEE Transactions on Cybernetics, 2018, (99):1-12.

Babak Pourasghar, Habib Izadkhah, Ayaz Isazadeh, Shahriar Lotfi, A graph-based clustering algorithm for software systems modularization, Information and Software Technology, 2020, 106469,

Spiros Mancoridis, Brian S. Mitchell, C. Rorres, Yih-Farn Chen, and Emden R. Gansner. Using automatic clustering to produce high-level system organizations of source code. InInternational Workshopon Program Comprehension (IWPC'98), pages 45–53, Los Alamitos, California, USA, 1998. IEEE Com-puter Society Press

D. Doval, S. Mancoridis, and B. S. Mitchell. Automatic clustering of software systems using a ge-netic algorithm. InInternational Conference on Software Tools and Engineering Practice (STEP'99),Pittsburgh, PA, 30 August - 2 September 1999.

M.R. Garey and D.S. Johnson.Computers and intractability. A Guide to the theory of NP-completeness.WH Freeman and Company, New York, 1979

K. Mahdavi, M. Harman, and R.M. Hierons, "A Multiple HillClimbing Approach to Software Module Clustering,"Proc. IEEEInt'l Conf. Software Maintenance,pp. 315-324, Sept. 2003.

Babak Pourasghar, Habib Izadkhah, Ayaz Isazadeh, Shahriar Lotfi, A graph-based clustering algorithm for software systems modularization, Information and Software Technology, 2020, 106469,

Jinhuang Huang, Jing Liu, A similarity-based modularization quality measure for software module clustering problems, Information Sciences, Volume 342, 2016, Pages 96-110,

Amarjeet prajapati, Jitender Kumar Chhabra, "Harmony Search Based Remodularization for Object-Oriented Software Systems", Computer Languages, Systems & Structures Volume 47, Part 2, January 2017, p 153–169

K. Praditwong, "Solving software module clustering problem by evolutionary algorithms," 2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE), Nakhon Pathom, 2011, pp. 154-159, doi: 10.1109/JCSSE.2011.5930112.

M. Harman, S. Swift, and K. Mahdavi, "An empirical study of the robustness of two module clustering fitness functions," in GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation. New York, NY, USA: ACM Press, 2005, pp. 1029-1036.

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

Sadat Jalali, N., Izadkhah, H. & Lotfi, S. Multi-objective search-based software modularization: structural and non-structural features. *Soft Comput* 23, 11141–11165 (2019). https://doi.org/10.1007/s00500-018-3666-z

Amarjeet prajapati, Jitender Kumar Chhabra, An empirical study of the sensitivity of quality indicator for software module clustering, IEEE Seventh International Conference on Contemporary Computing (IC3), 2014, p 206-211.

Amarjeet, Jitender Kumar Chhabra, "Improving Package Structure of Object-Oriented Software using Multi-objective Optimization and Weighted Class Connections ", Journal of King Saud University-Computer and Information Science, Volume 29, Issue 3, July 2017, p 349-364

Amarjeet prajapati, Jitender Kumar Chhabra, PSO-MoSR: A PSO based Multi-objective Software Remodularization", International Journal of Bio-Inspired Computation, 15 (4) 2020

A. Charan Kumari, K. Srinivas, Hyper-heuristic approach for multi-objective software module clustering,Journal of Systems and Software, Volume 117, 2016, Pages 384-401,

Marcio de Oliveira Barros. 2012. An analysis of the effects of composite objectives in multiobjective software module clustering. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation* (*GECCO '12*). Association for Computing Machinery, New York, NY, USA, 1205–1212.

Prajapati, A., Parashar, A. & Chhabra, J.K. Restructuring Object-Oriented Software Systems Using Various Aspects of Class Information. *Arab J Sci Eng* (2020). https://doi.org/10.1007/s13369-020-04785-z

Amarjeet Prajapati, Zong Woo Geem, Harmony Search-Based Approach for Multi-Objective Software Architecture Reconstruction. Mathematics, 2020, 8, 1906; doi:10.3390/math8111906

Yongqi Liu, Hui Qin, Zhendong Zhang, Liqiang Yao, Chao Wang, Li Mo, Shuo Ouyang, Jie Li, A region search evolutionary algorithm for many-objective optimization, Information Sciences, Volume 488, 2019, Pages 19-40,

Y. Sun, G. G. Yen and Z. Yi, "IGD Indicator-Based Evolutionary Algorithm for Many-Objective Optimization Problems," in *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 173-187, April 2019,

Yuan Liu, Ningbo Zhu, Kenli Li, Miqing Li, Jinhua Zheng, Keqin Li, An angle dominance criterion for evolutionary many-objective optimization, Information Sciences, Volume 509, 2020, Pages 376-399,

H. Chen, Y. Tian, W. Pedrycz, G. Wu, R. Wang and L. Wang, "Hyperplane Assisted Evolutionary Algorithm for Many-Objective Optimization Problems," in *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3367-3380, July 2020

L. Pan, L. Li, C. He and K. C. Tan, "A Subregion Division-Based Evolutionary Algorithm with Effective Mating Selection for Many-Objective Optimization," in *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3477-3490, Aug. 2020,

Y. Xiang, Y. Zhou, M. Li and Z. Chen, "A Vector Angle-Based Evolutionary Algorithm for Unconstrained Many-Objective Optimization," in *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 131-152, Feb. 2017

D. Gong, J. Sun and Z. Miao, "A Set-Based Genetic Algorithm for Interval Many-Objective Optimization Problems," in *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 47-60, Feb. 2018

Amarjeet prajapati, Zong Woo Geem, Harmony Search-Based Approach for Multi-Objective Software Architecture Reconstruction. Mathematics (MDPI), 2020, 8, 1906; doi:10.3390/math8111906

Sahar Saeedi, Reihaneh Khorsand, Somaye Ghandi Bidgoli, Mohammadreza Ramezanpour, Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing, Computers & Industrial Engineering, Volume 147, 2020, 106649.

C. Yue, B. Qu, J. Liang, A multiobjective particle swarm optimizer using ring topology for solving multimodal multiobjective problems, IEEE Trans. Evol. Comput. 22 (5) (2018) 805–817

C.A.C. Coello, M.S. Lechuga, MOPSO: A proposal for multiple objective particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Vol. 2, IEEE, 2002, pp. 1051–1056.

S. Zapotecas Martínez, C.A. Coello Coello, A multi-objective particle swarm optimizer based on decomposition, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation - GECCO '11, ACM Press, New York, USA, 2011, pp. 69–76.

N. Al Moubayed, A. Petrovski, J. McCall, D2 Mopso: MOPSO based on decomposition and dominance with archiving using crowding distance in objective and solution spaces, Evol. Comput. 22 (1) (2014) 47–77.

C. Dai, Y. Wang, M. Ye, A new multi-objective particle swarm optimization algorithm based on decomposition, Inform. Sci. 325 (2015) 541–557.

E.M.N. Figueiredo, T.B. Ludermir, C.J.A. Bastos-Filho, Many Objective Particle Swarm Optimization, Information Sciences, Volume 374, 2016, Pages 115-134,

W. Hu, G. G. Yen and G. Luo, "Many-Objective Particle Swarm Optimization Using Two-Stage Strategy and Parallel Cell Coordinate System," in *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1446-1459, June 2017,

Q. Lin *et al*., "Particle Swarm Optimization With a Balanceable Fitness Estimation for Many-Objective Optimization Problems," in *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 32-46, Feb. 2018.

Li Li, Guangpeng Li, Liang Chang, A many-objective particle swarm optimization with grid dominance ranking and clustering, Applied Soft Computing, Volume 96, 2020, 106661,