Hybrid of genetic algorithm and krill herd for software clustering problem

Mehdi Akbari
Department of Computer Engineering
University College of Daneshvaran
Tabriz, Iran
mat.akbari83@gmail.com

Habib Izadkhah
Department of Computer Science
University of Tabriz
Tabriz, Iran
izadkhah@tabrizu.ac.ir

***Abstract*— Clustering techniques are usually utilized to partition a software system, aiming to understand it. Understanding a program helps to maintain the legacy source code. Since the partitioning of a software system is an NP-hard problem, using the evolutionary approaches seems reasonable. Krill herd (KH) evolutionary algorithm is an effective algorithm for solving optimization problems with continuous state space which imitates the individual and group behavior of krill. According to its nature, is unable to solve the discrete space problems. The main advantage of this algorithm is to keep the information flow between different individuals during the evolutionary process. Genetic algorithm (GA) is an evolutionary algorithm utilizing the search techniques to find the closest solution to optimal; however, its main problem is a lack of strong effective information flow between different generations. This paper proposes a new evolutionary method, named GAKH, for software clustering inspired by Krill herd and Genetic algorithm. In the proposed evolutionary algorithm, the strengths of these two algorithms have been utilized and better results have been achieved in software clustering by changing GA cycle and operators, adding swarm intelligence and inspiring from Krill movements. The initial results achieved from the application of the proposed algorithm on ten software systems indicate higher quality results of clustering compared to other algorithms.**

Keywords— software clustering, genetic algorithm, Krill Herd, swarm intelligence

I. INTRODUCTION

In the modern era, the development of the internet and networks has made individuals, organizations and governments dependent on software. They are expanded and developed proportionately to demands of their users and the need to develop and maintain these software systems have turned into an important issue in research and development which necessitates the perception of a program for its development. As an example, according to one study [1], 40% to 60% of the software maintenance activities are allocated to its perception. The perception of a program is possible through the extraction of software architecture which shows its components and their relations. Software clustering is a method for architectural analysis and understanding of software structure, development, maintenance and test of software and clarifies its internal relations. Software clustering techniques are used to extract the software architecture from source code with the aim of perception and maintenance of the software [1, 2]. In fact, software clustering is the most common and popular method for discovery of structure and interrelations of large software. The main objective of clustering is making proper clusters and placing any artifact in a proper one based on some criteria including distance and similarity. Clustering is physical structure of software system, in fact, in a software system, the classes constitute the logical structure of a system and by grouping the logically related classes in a cluster, the physical structure of a system becomes clear and modifiable if required. The input of a clustering algorithm is an artifact dependency graph (ADG), whose nodes indicating some artifacts (such as class, file, method, etc.) and edges indicating the type of communications between artifacts (such as calling, inheritance, etc.). Figure 1 displays the clustering process for a software system with eight artifacts.

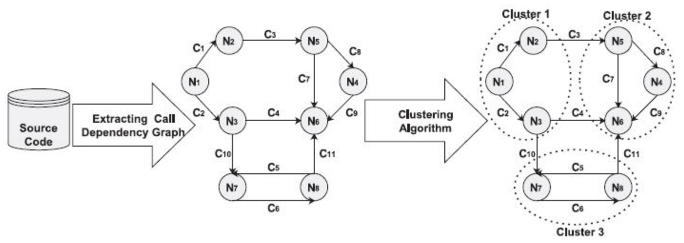

Fig. 1. The process of software system clustering [2]

Software clustering has been used in two different fields in software engineering including Software Architecture Recovery and Refactoring. The objective of software architecture recovery is the use of clustering techniques for analysis of a software system from the source code to the meaningful and understandable subsystems. In fact, it helps the reverse engineering process; however, refactoring utilizes clustering with different objectives. It supports big changes in re-modularization, i.e. a system is taken as input and presented as the output of a completely new system with new and improved modularization.

In the literature, two methods have been presented for software clustering including hierarchical and search-based methods. Failure to perform a full search of the problem and lack of appropriate criteria for cutting clustering are the reasons for the low quality of hierarchical clustering methods, which made them rarely used nowadays in clustering. On the other hand, due to their good tree structure, understanding a software through these methods is simple and easy. However, the search-based methods are more popular due to the acceptable quality of achieved clusters. GA is one of the efficient methods for clustering with a capability of the global search for software clustering problem; although some problems such as lack of efficient information flow between population (i.e. lack of strong communication and exchange of information between generations) sometimes yield disappointing results.

KH is a meta-heuristic algorithm with swarm intelligence presented for solving problems with continuous space [3]. One of the main advantages of this evolutionary algorithm is strong information flow between its components. One of the main components of swarm intelligence is information flow whose essential is the exchange of information for communication and cooperation between components in population and the other is self-organization, i.e. all components fulfill their tasks according to regulation. In fact, adding swarm intelligence (effective information flow and self-organization) to GA algorithm enables it to purposefully communicate and compete with its past and other components.

Software clustering is a problem in discrete space. This paper aims to present a new algorithm for software clustering by imitation of KH, discretization of its operators and its combination with genetic algorithm, which considerably improves the results by overcoming the weaknesses of these two algorithms. The contributions of this paper are summarized as follow:

1. Discretizing the KH with respect to the nature of the software clustering problem,
2. Addition of swarm intelligence to the genetic algorithm,
3. Presenting a new hybrid meta-heuristic through a combination of KH and GA.

The remainder of this paper is organized as follows. Section 2 presents a brief review of GA and KH algorithms. The proposed method is presented in Section 3. In Section 4, we evaluate the proposed algorithm on ten software systems. Finally, Section 4, concludes the paper.

II. RELATED WORKS

This section is divided into two sections: basic concepts and previous solutions.

A. Basic concepts

In this subsection, the basic concepts about GA and KH algorithms are expressed.

Genetic Algorithm. Genetic algorithm belongs to a collection of evolutionary algorithms which are based on biological evolution principles. In this algorithm, the fitness of a population improves over time through genetic operators including mutation and cross-over and selection of appropriate population in each generation. In overall, it could be claimed that genetic algorithm is based on an initial population of individuals (chromosomes) and each individual is a solution of the problem; these individuals change in consecutive iterations by genetic operators and approaches the optimum solution. Figure 2 presents a genetic algorithm steps.

```
Input: population size, crossover probability, mutation probability, generation numbers
 1. [Start] Generate population with random candidate solutions (the population size depends on the nature of the problem)
 2. [Fitness] Evaluate the fitness of each candidate in the population
 3. [New population] Create a new population by repeating following steps until the new population is complete
     1. [Selection] Select two parent candidates from a population according to their fitness (the better fitness, the bigger chance to be selected)
     2. [Crossover] With a crossover probability cross over the parents to form a new offspring (children).
     3. [Mutation] With a mutation probability mutate new offspring at each locus (position in candidate).
     4. [Accepting] Place new offspring in a new population
   4. [Replace] Select new generated population for the next generation
   5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
   6. [Loop] Go to step 2
```

Fig. 2. Genetic algorithm steps

Krill herd algorithm. KH is an effective meta-heuristic that was presented by Amir Gandomi in 2013 [3]. KH algorithm imitates the behavior of krill swarm during the search for food. Each krill individual is affected by three factors: (i) movement induced by the other individuals (ii) foraging activity and (iii) random diffusion [3]. In induced movement, each krill attempts to maintain the highest density. Two factors are effective in foraging activities: where the food is located, and where the previous food was located. Random diffusion is expressed according to the maximum propagation speed and a random directed vector. The KH algorithm steps are as follow [3]:

1. Data Structures: Define the simple bounds, determination of algorithm parameter(s), etc.

2. Initialization: Randomly create the initial population in the search space.
3. Fitness evaluation: Evaluation of each krill individual according to its position.
4. Motion calculation:
   a. Motion-induced by the presence of other individuals,
   b. Foraging motion
   c. Physical diffusion
5. Implement the genetic operators
6. Updating: updating the krill individual position in the search space.
7. Repeating: go to step 3 until the stop criterion is reached.
8. End.

B. *software clustering algorithms*

In this subsection, some software clustering algorithms are studied.

Bunch algorithm. The length of the chromosome in bunch depends on the number of nodes of the graph and the content of each gene indicates the cluster in which the node is located [4].

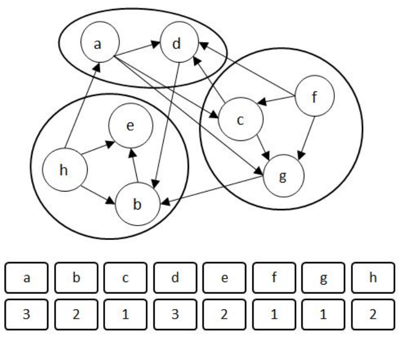

Fig. 3. Bunch algorithm encoding

DAGC algorithm. Let 'm' in DAGC indicates the chromosome number and 'p' is the number of nodes of the graph produced through permutation, if 'm <= p', then 'm' will be placed in a new cluster; otherwise, it belongs to 'p' cluster [5].

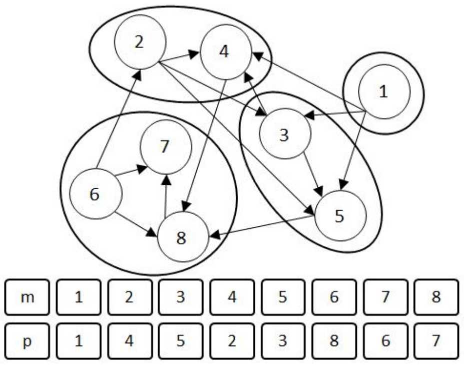

Fig. 4. DAGC algorithm encoding

Hill climbing algorithm. This algorithm always selects the point with more value, i.e. the closest point to the top of the hill. In other words, initiating from a random point and creating a neighbor for it, this algorithm selects the neighbor with the highest quality and when reached the top of the hill, it will terminate; moreover, if the quality of neighbor is lower, the algorithm will terminate [6]. Hence, two hill-climbing-based algorithms for software clustering, namely, Steepest Ascent Hill-climbing (SAHC) and Next Ascent Hill-climbing (NAHC), were proposed by Mitchell in [6]. In NAHC, the algorithm stops searching for a neighbor when the first neighboring partition with a higher objective function is found and then repeats this process for that neighbor. In SAHC, the algorithm stops searching for a neighbor when it has examined all neighboring partitions. It then chooses the one with the largest objective function and repeats this process for the neighbor found. Table I shows a number of software clustering algorithms.

TABLE I. SOFTWARE CLUSTERING ALGORITHMS.

Algorithm	Algorithm type
Software clustering using Hill climbing algorithm [6]	Local search based
Software clustering using SA algorithm [4]	Local search based
Software clustering using Multiple hill climbing [7]	Local search based
Software clustering using E-CDGM [8]	Local search based
Local search on software clustering [9]	Local search based
Semantic Clustering with Hill climbing [10]	Local search based
BUNCH [4, 6]	Global search based
DAGC [5]	Global search based
Software clustering using firefly algorithm [11]	Global search based
ECA [12]	Global search based
MCA [12]	Global search based
Software clustering by combining evolutionary-social algorithms [13]	Global search based
Harmonic search based clustering for object oriented systems [14]	Global search based
GA-SMCP [15]	Global search based
Multi-agent evolutionary algorithm [16]	Global search based
Software clustering by black hole algorithm [17]	Global search based
EDA [18]	Global search based
Software clustering by new hybrid algorithms [19]	Local and global search based

III. THE PROPOSED ALGORITHM

In the classic GA algorithm, concerning the lack of strong information flow between parents and children and between different generations, we witness failure in the achievement of the best results in some problems. In the proposed method, we tried to considerably improve the software clustering results according to the presented results using combined GA and KH algorithms and imitation of krill movement models in GA operators and introduce a new method for solving clustering problems using KH algorithm.

The main reason for proposing this change was that GA algorithm did not yield the best results compared to other algorithms such as KH due to lack of strong information flow; on the other hand, KH genetic operators did not considerably improve the results. This algorithm proposed to overcome the weaknesses of KH and GA and solving the clustering problems by KH. Results section demonstrates that the obtained results are considerably better than previous results.

The steps of the proposed algorithm are as follows.

1. Initialization: Randomly create the initial population in the search space.
2. Fitness evaluation: Evaluating the fitness of each candidate (chromosome) by TurboMQ.
3. Motion Operators:
   a. Cumulative Motion
   b. Local Motion
   c. Random Motion
4. Replacement
5. Update population
6. Repeat: go to step 3 until the stop criteria is reached
7. End.

In the following, we explain the steps 3 and 4 of the proposed algorithm.

Step 3: Fitness evaluation by TurboMQ. To measure the quality of clustering, TurboMQ is used [4, 6]. To calculate TurboMQ, first, the module factor (CF) must be calculated per module where this value is obtained through the internal and external relations of clusters as Eq. 1 and Eq. 2.

$$\text{TurboMQ} = \sum_{i=1}^{k} \text{CF}_i \tag{1}$$

$$\text{CF}_i = \begin{cases} 0 & \text{if } \mu_i = 0 \\ \frac{2\mu_i}{2\mu_i + \varepsilon_i} & \text{otherwise} \end{cases} \tag{2}$$

In this equation, μ_i and ε_i denote the total number of internal and external relations for module i, respectively. The higher TurboMQ indicate the better the internal relations between the modules.

Step 4: in this step of the algorithm, by using the GA operators and adding self–organization, parent information flow, the best and worst objective functions as well as results of the previous generations to these operators and changing the population creation process in GA, the resulting generations provide better results than the previous generations. In fact, each chromosome imitating the krill's behavior and through (i) cumulative, (ii) local, and (iii) random movements proceeds towards the best results in each generation.

Cumulative movement. This movement is inspired by the krill induced movement. Krill in this movement tries to maintain the highest density and to get closer to the group's largest gathering [5]. We simulate this movement in this paper as follows.

Let operator ⊗ indicates the cross-over as that it takes some parent chromosomes as inputs and produces two children as outputs and let operator ⊖ indicates cross-over operator as that it takes two parent chromosomes as inputs and produces a child as output. Operator "+" represents the selection of a higher quality chromosome among the existing chromosomes. Let ch_i, ch_j, ch_{best}, ch_{worst}, $ch_{i,best}$, $ch_{i,worst}$ indicates, respectively, the parent chromosome 1, the parent chromosome 2, the best chromosome in the generation (the best global), the worst chromosome in the generation (the worst global), the best past of chromosome i (the best local) and the worst past of chromosome i (the worst local). To carry out this movement, we define the following relationships.

The cross-over operator is applied on two chromosomes i and j with respect to Eq. 3.

$$x_{i,j} = \begin{cases} x_{ofspring1,2}, & \text{if rand}_{i,j} < \text{Cr} \\ x_{i,j} & \text{else} \end{cases} \tag{3}$$

where rand is between [0, 1] and Cr is the cross-over rate.

The cross-over operator is applied on three chromosomes i, j and k with respect to Eq. 4.

$$x_{i,j,k} = \begin{cases} x_{ofspring1,2,3}, & \text{if rand}_{i,j,k} < \text{Cr} \\ x_{i,j,k} & \text{else} \end{cases} \tag{4}$$

where rand is between [0, 1] and Cr is the cross-over rate.

$$X = (ch_i \otimes ch_j) + (ch_{best} \otimes ch_{worst}) \tag{5}$$

$$Y = (ch_i \otimes ch_j \otimes ch_{best}) + (ch_i \otimes ch_j \otimes ch_{worst}) \tag{6}$$

$$A = (ch_i \ominus ch_{best}) + (ch_i \ominus ch_{worst}) \tag{7}$$

$$B = (ch_j \ominus ch_{best}) + (ch_j \ominus ch_{worst}) \tag{8}$$

$$K = (ch_i \otimes ch_{i,best}) + (ch_j \otimes ch_{j,best}) \tag{9}$$

$$Z = (ch_i \otimes ch_{i,worst}) + (ch_j \otimes ch_{j,worst}) \tag{10}$$

$$N = \alpha X + \beta Y + \gamma A + \delta B + \eta K + \vartheta Z \tag{11}$$

where N represents results of the cumulative movement and the binary coefficients indicate selection and the lack of selection of the items.

Local movement. This move is inspired by the foraging movement of krill. In this move, krill knowing the previous place of food looks for food [5]. To simulate this move, we have defined the following relation. Operator mutation as ⊕ takes some chromosomes as inputs and produces a new chromosome using Eq. 13.

$$ch_{i,j} = \begin{cases} ch_{r,j} & \text{if rand}_{i,j} < \text{Mu} \\ ch_{i,j} & \text{else} \end{cases} \tag{12}$$

$$F = ch_i \oplus ch_{best} \oplus ch_{i,best} \tag{13}$$

where Mu is the mutation rate.

Random diffusion. This move is inspired by the random movement of the krill. In this move, the proposed algorithm in each generation produces D^{max} new chromosomes and adds them to the generation. Then, chromosomes are selected based on their quality and the population size. The important point is that the number of random movements during the evolutionary process of the algorithm based on the following equation decreases using Eq. 14.

$$D_i = D^{max}\left(1-\frac{I}{I_{max}}\right) \qquad (14)$$

where i represents the generation number and I_{max} represents the maximum number of generations. The value of i is an input parameter and based on experiments is set to 0.3.

IV. RESULTS

For evaluating the proposed algorithm, it applied to a number of software systems. The number of iterations for all algorithms are considered the same. All algorithms are executed 20 times and the best results have been selected from different independent runs. Cross-over rate is considered as 0.8 and mutation rate as 0.15 Table II presents the initial results of comparing GAKH proposed algorithm with some evolutionary algorithms on ten software systems. As seen in this table, in all cases, the achieved quality by the proposed algorithm is better than or equal to the compared algorithms.

TABLE II. TURBOMQ FOR GAKH AND SIX KNOWN SEARCH-BASED

	GAKH	Bunch	DAGC	NAHC	SAHC	E-CDGM	EDA
Compiler	**1.506**	**1.506**	**1.506**	1.142	1.142	**1.506**	**1.506**
Boxer	**3.101**	**3.101**	**3.101**	1.711	1.633	2.87	2.91
Acqcigna	**16.602**	7.225	0.898	5.740	6.049	1.912	4.099
Ispell	**2.189**	2.177	1.997	1.416	1.416	1.416	2.177
cia++	**15.687**	7.066	0.953	3.574	3.868	3.976	3.976
Cia	**2.790**	2.706	1.833	2.586	2.781	2.781	2.768
Ciald	**2.851**	**2.851**	2.463	1.7	1.715	1.915	**2.851**
Bison	**2.648**	2.606	1.763	1.442	1.437	1.220	1.763
Rcs	**2.187**	2.175	1.894	1.208	1.207	1.207	2.178
Start	**3.805**	**3.805**	0.845	2.155	2.195	2.199	3.723

Figure 5 and 6 show the convergence chart for ispell and bison applications, respectively. These figures demonstrate the quick convergence of the proposed algorithm. Because evolutionary algorithms have a stochastic property, their stability assessment is important. Stability means that an algorithm in the various runs gives close solutions. Figure 7 and 8 show the stability chart for ispell and bison software systems on ten independent runs, respectively. In these figures, it is clear that the proposed algorithm has acceptable stability.

V. CONCLUSION

This paper proposed a new evolutionary algorithm inspired by the behavior of the genetic algorithm and swarm behavior of the KH algorithm for software clustering problem. The proposed algorithm was applied to a number of software systems. The initial results of the evaluation indicated the effectiveness of the proposed algorithm. The main advantage of the proposed algorithm is the use of information flow between different generations and having a good local and global search.

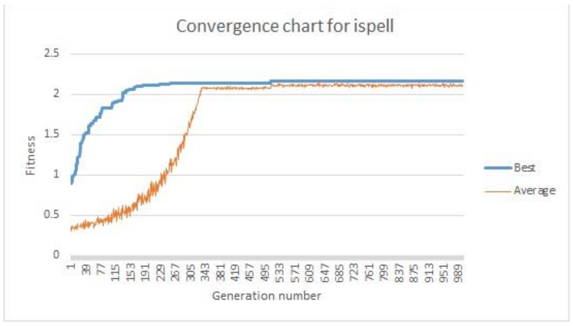

Fig. 5. Convergence for ispell

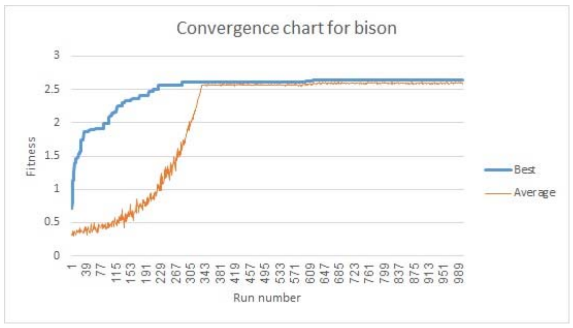

Fig. 6. Convergence for bison

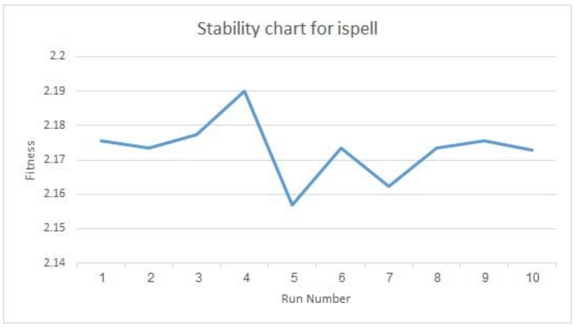

Fig. 7. Stability for ispell

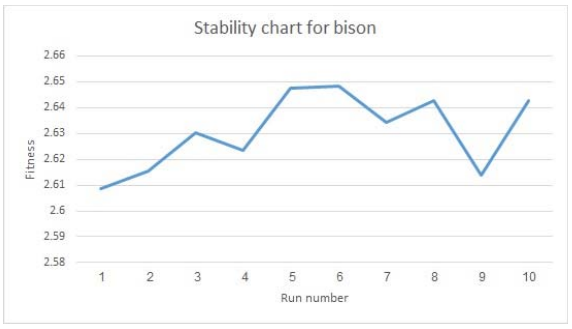

Fig. 8. Stability for bison

REFERENCES

[1] Beck, Fabian, and Stephan Diehl. On the impact of software evolution on software clustering. Empirical Software Engineering 18.5 (2013): 970-1004.

[2] Isazadeh, Ayaz, Habib Izadkhah, and Islam Elgedawy. *Source Code Clustering: Theory and Techniques*. Springer, 2017.

[3] Gandomi, Amir Hossein, and Amir Hossein Alavi. Krill herd: a new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation* 17, no. 12 (2012): 4831-4845.

[4] Mitchell, Brian S. A heuristic search approach to solving the software clustering problem. Ph.D. Theses. Drexel University, 2002.

[5] Parsa, Saeed, and Omid Bushehrian. A new encoding scheme and a framework to investigate genetic clustering algorithms. Journal of Research and Practice in Information Technology 37.1 (2005): 127.

[6] Mitchell, Brian S., and Spiros Mancoridis. On the automatic clustering of software systems using the bunch tool. IEEE Transactions on Software Engineering 32.3 (2006): 193-208.

[7] Mahdavi K. A clustering genetic algorithm for software modularisation with a multiple hill climbing approach (Doctoral dissertation, Brunel University).

[8] Izadkhah, Habib, Islam Elgedawy, and Ayaz Isazadeh. E-CDGM: An Evolutionary Call-Dependency Graph Clustering Approach for Software Systems. Cybernetics and Information Technologies 16, no. 3 (2016): 70-90.

[9] Monçores MC, Alvim AC, Barros MO. Large Neighborhood Search applied to the Software Module Clustering problem. Computers & Operations Research. 2018 Mar 1;91:92-111.

[10] Kargar, Masoud, Ayaz Isazadeh, and Habib Izadkhah. Semantic-based software clustering using hill climbing. In Computer Science and Software Engineering Conference (CSSE), 2017 International Symposium on, pp. 55-60. IEEE, 2017.

[11] Mamaghani, Ali Safari, and Meysam Hajizadeh. Software clustering using the modified firefly algorithm. In Software Engineering Conference (MySEC), 2014 8th Malaysian, pp. 321-324. IEEE, 2014.

[12] Praditwong, Kata, Mark Harman, and Xin Yao. Software module clustering as a multi-objective search problem. IEEE Transactions on Software Engineering 37, no. 2 (2011): 264-282.

[13] Jeet, Kawal, and Renu Dhir. "Software Module Clustering Using Hybrid Socio-Evolutionary Algorithms." International Journal of Information Engineering and Electronic Business 8, no. 4 (2016): 43.

[14] Chhabra JK. Harmony search based reclustering for object-oriented software systems. Computer Languages, Systems & Structures. 2017 Jan 1;47:153-69.

[15] Huang, Jinhuang, and Jing Liu. A similarity-based clustering quality measure for software module clustering problems. Information Sciences 342 (2016): 96-110.

[16] Huang, Jinhuang, Jing Liu, and Xin Yao. A multi-agent evolutionary algorithm for software module clustering problems. *Soft Computing* 21, no. 12 (2017): 3415-3428.

[17] Jeet, Kawal, and Renu Dhir. Software architecture recovery using genetic black hole algorithm. ACM SIGSOFT Software Engineering Notes 40, no. 1 (2015): 1-5.

[18] Tajgardan, Mahjoubeh, Habib Izadkhah, and Shahriar Lotfi. Software Systems Clustering Using Estimation of Distribution Approach. Journal of Applied Computer Science Methods 8, no. 2 (2016): 99-113.

[19] Mamaghani AS, Meybodi MR. Clustering of software systems using new hybrid algorithms. InComputer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on 2009 Oct 11 (Vol. 1, pp. 20-25). IEEE.