Chinese Journal of Electronics
Vol.32, No.1, Jan. 2023

Developer Cooperation Relationship and Attribute Similarity Based Community Detection in Software Ecosystem

SHEN Xin[1], DU Junwei[2], GONG Dunwei[1], and YAO Xiangjuan[3]

(1. *School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China*)
(2. *School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, China*)
(3. *School of Mathematics, China University of Mining and Technology, Xuzhou 221116, China*)

Abstract — A software ecosystem (SECO) can be described as a special complex network. Previous complex networks in an SECO have limitations in accurately reflecting the similarity between each pair of nodes. The community structure is critical towards understanding the network topology and function. Many scholars tend to adopt evolutionary optimization methods for community detection. The information adopted in previous optimization models for community detection is incomprehensive and cannot be directly applied to the problem of community detection in an SECO. Based on this, a complex network in SECOs is first built. In the network, the cooperation intensity between developers is accurately calculated, and the attribute contained by each developer is considered. A multi-objective optimization model is formulated. A community detection algorithm based on NSGA-II is employed to solve the above model. Experimental results demonstrate that the proposed method of calculating the developer cooperation intensity and our model are advantageous.

Key words — Software ecosystem, Complex network, Cooperation intensity, Attribute similarity, Community detection.

I. Introduction

A software ecosystem (SECO) is a complex system formed by the interaction of software products, services, data, and knowledge among stakeholders such as software companies, developers, and technique communities in a specific technique environment [1]. Since its inception, an SECO has attracted great attention from academic and industrial communities.

As the highest level of current software engineering, an SECO usually has a high reliability and scalability, and evolves continuously with changes in demand. In an SECO, various people are generally involved, and they have complex relationships. Efficiently mining relationships among them is beneficial to understand the structure and behavior of the SECO, thereby promoting the healthy and sustainable development of the system. Community detection in complex networks has been a hot research topic in recent years [2]–[7], and community detection in an SECO can help us master its behavior, evolution and development, which is of great theoretical and practical significance. In view of this, we study the problem of community detection in an SECO.

In an SECO, the prerequisite for community detection is to build a complex network that reflects different entities and their relationships. Ref.[8] built five kinds of complex networks based on relationships between developers, between repositories, between developer and repository, and between user and repository, respectively. Further, Ref.[9] calculated the relationship intensity between developers. However, it is expected to further improve in matching common sense and containing more information. In addition, interaction behavior between developers is hoped to fully explore.

Based on this, we present a novel method of calculating the developer cooperation intensity in an SECO. The main idea of our method is shown as follows. According to different categories of developer cooperation

Manuscript Received Aug. 6, 2021; Accepted Apr. 12, 2022. This work was supported by the National Key Research and Development Program of China (2018YFB1003802).
© 2023 Chinese Institute of Electronics. DOI:10.23919/cje.2021.00.276

relationships, the developer cooperation intensity based on the network topology is calculated. The developer cooperation intensity based on the interaction behavior is measured according to the number of commitments of shared repositories and the times of cooperation between developers. The network topology and the interaction behavior between developers are integrated to obtain the developer cooperation intensity. It is easy to know that if developers contain the same attribute together, they often have similar backgrounds. If these developers are accurately identified, it will help them interact further. Therefore, we integrate the attribute contained by each developer into the SECO network, which makes the network information more comprehensive.

Since Ref.[10] proposed the concept of community detection in complex networks, previous community detection approaches can be roughly divided into the following three categories. The first category is network segmentation. The representative method is the Kernighan-Lin algorithm [11]. With respect to the second category, it is hierarchical clustering. For this category, it is mainly divided into a split method and an aggregation method. The former performs community detection from top to bottom such as the Girvan-Newman algorithm [10], and the latter detects communities from bottom to top, like the FastNewman algorithm [12] and the Louvain algorithm [13]. Regarding the third category, it is based on evolutionary optimization. Representative evolutionary optimization methods are genetic algorithm [14], particle swarm optimization [15], etc. Compared to the first two categories, the third one is relatively simpler and has better performance in robustness. However, it needs an optimization model for community detection. At present, previous optimization models have mainly considered the structure relationship of a network [14], [16]–[18]. In addition, scholars have constructed optimization models based on the network structure and attribute similarity for community detection [19], [20]. However, the information adopted in these models is not comprehensive and cannot be directly applied to the problem of community detection in an SECO.

In view of this, we are the first to establish an optimization model for community detection in an SECO. This model considers not only the developer cooperation intensity in a community, but also the programming language similarity between developers in a community. Additionally, when calculating the developer cooperation intensity in a community, the direct and indirect cooperation intensities as well as interaction behavior between developers are adopted to make the information used in the optimization model more comprehensive. In this way, communities with close developer cooperation and high programming language similarity can be obtained.

This paper has the following four-fold contributions:

1) Presenting a novel method of calculating the developer cooperation intensity in an SECO. This method comprehensively and accurately measures the developer cooperation intensity from two aspects, that is, the network topology and interaction behavior.

2) Considering the attribute contained by each developer into the formed networks. The information contained in these network is more comprehensive, which lays a foundation for community detection.

3) Formulating an optimization model for community detection in the formed networks. The two objectives in the optimization model are constructed, by taking both the developer cooperation relationship and the attribute similarity into consideration, which is helpful to obtain a community division with good performance.

4) Experimentally investigating the effectiveness of the proposed method and model. Complex networks in a software ecosystem are built using data crawled in GitHub. The results show that the community structure obtained by the proposed community detection method is the most significant when the network topology and interaction behavior between developers are considered equally important in calculating the developer cooperation intensity, and the proposed method of calculating the developer cooperation intensity performs better than the other two methods in most cases. Moreover, our model is beneficial to obtain a more satisfactory community division compared to the other five models.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III gives an overall framework of constructing the SECO network, and presents a method of calculating the developer cooperation intensity. An optimization model for community detection in the SECO is formulated in Section IV. Section V is an evolutionary optimization method to solve the formulated optimization model. Section VI verifies the effectiveness of the built network and the optimization model through comparative experiments. Finally, Section VII concludes the whole paper, and points out the topics to be studied in the future.

II. Related Work

This section reviews previous studies from the following two aspects, and software ecosystem, community detection. Based on this, the motivation of this paper is specified.

1. Software ecosystem

Since the concept of an SECO is proposed [1], it

has been widely concerned by the academia and industry. SECO is deeply integrated with emerging information technologies such as open-source software, mobile applications, and cloud computing, forming open-source SECO, cloud computing SECO, mobile application SECO. Further, more and more enterprises have begun to use SECO platforms to build software, such as Apples' IOS and Googles' Android Smartphone ecosystem [21].

For an SECO, its reliability and robustness has attracted great attention from different participants, such as software developers, software users, software companies, etc. Ref.[22] proposed various evaluation indicators to measure the performance of R ecosystem. Ref.[23] proposed a maturity model of SECOs governance, so as to determine on whether to invest in an ecosystem for stakeholders. In addition, Ref.[24] conducted an empirical study on cross-project-related errors in the Python SECO, revealing common practices of developers and various factors for repairing cross-project errors.

Essentially, an SECO is a complex network, where the community structure is an essential topology feature. So far, the existing complex networks have been built according mainly to the relationship between entities, ignoring attributes owned by these entities. In addition, strong or weak relationships between entities have not been fully utilized. Although some studies are associated with the above relationships, the accuracy of calculating the relationship intensity is expected to further improve.

2. Community detection

As mentioned earlier, previous community detection methods of complex networks are mainly divided into network segmentation, hierarchical clustering, and evolutionary optimization. Among them, the prerequisite of network segmentation is to know the number of communities and that of nodes contained in each community. However, for an actual complex network, it is usually difficult to meet this prerequisite. In this situation, it is a feasible way to adopt hierarchical clustering and evolutionary optimization to detect communities in a complex network.

For hierarchical clustering, Ref.[25] introduced a division-and-agglomeration algorithm, in which a two-stage strategy is employed to achieve community detection in social networks. For similar complex networks, Ref.[26] proposed a three-stage algorithm which includes central node identification, label propagation and community combination stages.

Previous studies mainly take the structure of a complex network when formulating the optimization model. Ref.[16] took the modularity density as the objective to be optimized when formulating the optimization model for community detection. In the optimization model [14], multiple objectives are formed by minimizing the kernel k-means and the ratio cut. Ref.[17] took the conductance, the normalized cut and the community score as the objectives of the optimization problem. In addition, previous studies have formulated the optimization model for community detection based on the structure of a complex network and the node attribute similarity. Ref.[19] defined a function to reflect the node attribute similarity, and formulated the optimization objectives together with modularity [27]. It is worth noting that most optimization models for community detection take the network structure into consideration. In addition, some optimization models focus on the network structure and the node attribute, they are, however, mainly suitable for unweighted networks.

After establishing the optimization model of an optimization problem, it is critical to design problem-oriented optimization algorithms. Ref.[28] proposed a cooperative co-evolutionary algorithm for module identification, with its application to discover cancer diseases. Ref.[19] presented an evolutionary optimization algorithm for community detection through hybrid representation, improved evolutionary operators, and integrated local search. This algorithm adopts the non-dominated sorting genetic algorithm (NSGA-II) as a framework [29], which is the most representative multi-objective optimization algorithm.

To summarize, previous studies have some shortcomings in building networks and detecting communities that comprehensively and accurately reflect the developer cooperation relationship in an SECO. In view of this, this paper builds a complex network based on the developer cooperation relationship and attribute. Based on this, a bi-objective optimization model is formulated by considering the developer cooperation relationship and attribute similarity. Further, NSGA-II incorporating specific evolutionary operators is employed to solve our model. As a result, various schemes for community division are obtained.

III. Constructed Software Ecosystem Network

1. Overall framework

In this paper, an SECO network based on the cooperation relationship and programming language is built. This network is denoted as $G = (V, E, W, A)$ ($|V| = n, |E| = m$), where V represents the set of nodes, E represents the set of edges, W means the set of edge weights, and A refers to the set of attributes. The meanings of a node, a node attribute, an edge and its weight of the network are provided as follows.

1) A node of the network corresponds to a developer.

2) A node attribute of the network refers to a programming language mastered by a developer. A developer generally masters one or more programming languages.

3) An edge of the network corresponds to a joint committing of two developers to a repository.

4) An edge weight of the network is associated with the cooperation intensity between developers, with its initial value of 1. On this circumstance, $G=(V, E, A)$ is held.

Next, the method of calculating the developer cooperation intensity will be elaborated in detail.

2. Method of calculating developer cooperation intensity

If two or more developers commit the same repository, behaviors between these developers are interactive. Thus, the cooperation intensity between developers can be reflected through the network topology and interaction between developers. In view of this, we propose a method of calculating the developer cooperation intensity based on the network topology and interaction behavior in this section. The developer cooperation relationship is first classified based on the network topology, and the developer cooperation intensity is calculated according to different categories. Following that, the developer cooperation intensity is calculated based on interaction behavior, which contains the number of commitments of shared repositories and the times of cooperation between developers. Finally, the network topology and interaction between developers are comprehensively considered to measure the developer cooperation intensity.

1) Network topology based developer cooperation intensity

i) Classification of developer cooperation relationships

For two nodes, v_i and v_j, in the network, they have either a connection with an edge or not, and the distance between them is denoted as dis_{ij}. If the two nodes are connected, dis_{ij} will be equal to 1, and there will exist a direct cooperation relationship between their corresponding developers. If the two nodes have no connection but there is a path from v_i to v_j, dis_{ij} will be greater than 1 and be finite, and there will exist an indirect cooperation relationship between their corresponding developers. If there is no path from v_i to v_j, dis_{ij} will be infinity, and no cooperation relationship will exist between their corresponding developers. Therefore, from the perspective of cooperation, developers have the following three kinds of relationships, that is, direct cooperation, indirect cooperation, and no cooperation.

ii) Direct cooperation intensity between developers

For two developers, d_i and d_j, who have a direct cooperation relationship, d_i may have direct cooperation with other developers, and the same is true for d_j. The number of developers who have a direct cooperation relationship with d_i is denoted as $\alpha(d_i)$, which is equal to the degree of the corresponding node in the network. If $\alpha(d_i)$ is large, many developers will directly cooperate with d_i. Accordingly, the direct cooperation intensity between d_i and d_j will be small. The direct cooperation intensity between d_i and d_j is denoted as $S_1^{'}(d_i, d_j)$, with its expression being as follows.

$$S_1^{'}(d_i, d_j)=\frac{1}{\alpha(d_i)+\alpha(d_j)-1} \tag{1}$$

iii) Indirect cooperation intensity between developers

For two developers, d_i and d_j, who have an indirect cooperation relationship, they correspond to the two nodes, v_i and v_j, in the network, respectively. It is clear that the larger the value of dis_{ij}, the longer the distance between v_i and v_j, and the weaker their indirect cooperation. The indirect cooperation intensity between d_i and d_j is denoted as $S_2^{'}(d_i, d_j)$, with its expression being as follows.

$$S_2^{'}(d_i, d_j)=\frac{\mathrm{e}^{-dis_{ij}}}{\alpha(d_i)+\alpha(d_j)-1} \tag{2}$$

From (1) and (2), when d_i has an indirect cooperation relationship with d_j, dis_{ij} will be greater than 1. On this circumstance, one has $\mathrm{e}^{-dis_{ij}}<\mathrm{e}^{-1}<\mathrm{e}^{0}=1$, and thus $S_2^{'}(d_i, d_j)<S_1^{'}(d_i, d_j)$.

iv) No cooperation between developers

If the developer, d_i, has no cooperation with d_j, there will be no path from v_i to v_j. In this case, their cooperation intensity is equal to 0.

In summary, the cooperation intensity between d_i and d_j based on the network topology is denoted as $S^{'}(d_i, d_j)$, which can be expressed as follows.

$$S^{'}(d_i, d_j)=\begin{cases}\dfrac{1}{\alpha(d_i)+\alpha(d_j)-1}, & dis_{ij}=1\\ \dfrac{\mathrm{e}^{-dis_{ij}}}{\alpha(d_i)+\alpha(d_j)-1}, & dis_{ij}>1\end{cases} \tag{3}$$

2) Interaction behavior based developer cooperation intensity

Among various developer interactions, different developers sometimes commit the same repository with different number of commitments. The same repository is called a shared repository. For two developers, the number of commitments of their shared repositories can reflect their cooperation intensity. In addition, the times of developer cooperation have an influence on the

cooperation intensity between developers.

i) Developer cooperation intensity in terms of the number of commitments of shared repositories

For a shared repository, different developers have different number of commitments, and accordingly, they have different contributions to the shared repository. In view of this, the number of commitments of shared repositories can be employed to reflect a developerś contribution to them. Further, the cooperation intensity between developers can be characterized by their contributions to the shared repositories.

For two developers, d_i and d_j, the set of their shared repositories is denoted as P_{ij}, with its size of $|P_{ij}|$. Taking sth shared repository of d_i and d_j as an example, denoted as p_{ij}^s, the number of commitments of d_i to the repository is denoted as $N_i(p_{ij}^s)$, and that of d_j to the repository is denoted as $N_j(p_{ij}^s)$. Then, the times of cooperation of d_i and d_j to the repository are equal to $\min\left\{N_i(p_{ij}^s), N_j(p_{ij}^s)\right\}$, denoted as $N_{\min}(p_{ij}^s)$.

For p_{ij}^s, in addition to being a shared repository of d_i and d_j, the repository can be a shared repository of other developers. That is, it can be jointly committed by other developers. The number of commitments of p_{ij}^s is denoted as $N(p_{ij}^s)$. Then, the larger the ratio of $N_{\min}(p_{ij}^s)$ to $N(p_{ij}^s)$, the greater the joint contributions of d_i and d_j to p_{ij}^s among all developers. For all shared repositories in P_{ij}, the larger the above ratio, the greater the sum of joint contributions of d_i and d_j to them among all developers. If the two developers contribute more to their shared repositories, their cooperation will be stronger.

The cooperation intensity between d_i and d_j in terms of the number of commitments of their shared repositories is denoted as $S_1^{''}(d_i, d_j)$, which can be expressed as follows.

$$S_1^{''}(d_i, d_j) = \frac{1}{|P_{ij}|} \sum_{p_{ij}^s \in P_{ij}} \frac{N_{\min}(p_{ij}^s)}{N(p_{ij}^s)} \tag{4}$$

ii) Developer cooperation intensity in terms of the times of cooperation between developers

In an SECO network, the set of developers is denoted as $D = \{d_1, d_2, \ldots, d_n\}$. For the two developers, d_i and d_j, the times of their cooperation in p_{ij}^s are denoted as $N_{\min}(p_{ij}^s)$. Further, the times of their cooperation to all shared repositories are equal to $\sum_{p_{ij}^s \in P_{ij}} N_{\min}(p_{ij}^s)$, and the larger the value of $\sum_{p_{ij}^s \in P_{ij}} N_{\min}(p_{ij}^s)$, the larger their cooperation intensity.

The developer cooperation intensity in terms of the times of cooperation between the two developers d_i and d_j is denoted as $S_2^{''}(d_i, d_j)$, which can be calculated as follows.

$$S_2^{''}(d_i, d_j) = \frac{\sum_{p_{ij}^s \in P_{ij}} N_{\min}(p_{ij}^s)}{\max\limits_{d_k, d_l \in D} \sum_{p_{kl}^s \in P_{kl}} N_{\min}(p_{kl}^s)} \tag{5}$$

To sum up, the interaction behavior based developer cooperation intensity between d_i and d_j is denoted as $S^{''}(d_i, d_j)$, which has the following expression.

$$S^{''}(d_i, d_j) = \frac{1}{|P_{ij}|} \sum_{p_{ij}^s \in P_{ij}} \frac{N_{\min}(p_{ij}^s)}{N(p_{ij}^s)} \cdot \frac{\sum_{p_{ij}^s \in P_{ij}} N_{\min}(p_{ij}^s)}{\max\limits_{d_k, d_l \in D} \sum_{p_{kl}^s \in P_{kl}} N_{\min}(p_{kl}^s)} \tag{6}$$

3) Network topology and interaction behavior based developer cooperation intensity

The methods of calculating the developer cooperation intensity based on the network topology and interaction behavior have been presented. Given the fact that the developer cooperation intensity has a close relation with the two aspects, the developer cooperation intensity can be obtained by combining the above two cooperation intensities. The cooperation intensity between d_i and d_j is denoted as $S(d_i, d_j)$, which can be expressed as follows.

$$S(d_i, d_j) = \tau S^{'}(d_i, d_j) + (1 - \tau) S^{''}(d_i, d_j) \tag{7}$$

In this formula, τ is a weighted factor. If $S^{'}(d_i, d_j) > 0$ and $S^{''}(d_i, d_j) > 0$, the cooperation intensity between d_i and d_j will be called the developer direct interaction intensity, which corresponds to an edge weight of the network.

Seldom studies [9] are associated with comprehensively evaluating the developer cooperation intensity based on the network topology and interaction behavior. However, the developer cooperation intensity based on the network topology is expected to further improve in matching common sense. In addition, interaction behavior between developers is hoped to fully explore.

IV. Constructed Optimization Model

The community detection problem studied in this section can be described as follows. A number of communities in an SECO network are detected, so that developers corresponding to nodes in each community cooperate closely, and there is a high attribute similarity among them.

1. Decision variable

The set of nodes in a network is denoted as $V = \{v_1, v_2, \ldots, v_n\}$, the number of communities is de-

noted as I, and a division in a network is denoted as $X=\{X_1,X_2,\ldots,X_I\}$. For X_i, $i=1,2,\ldots,I$, one has $X_i\subseteq 2^V$. For X_i, X_j, $i,j=1,2,\ldots,I$, they should satisfy that $X_i\cap X_j=\emptyset$. Additionally, it is required that $\bigcup_{i=1}^{I} X_i=V$. When formulating the optimization model, the decision variable is chosen as X.

2. Objectives

1) Developer cooperation intensity in community

The number of nodes in community X_i is denoted as $|X_i|$. If $|X_i|$ is larger than 1, we will take the nodes, v_k^i and v_l^i, in X_i into consideration. The direct cooperation intensity between the developers d_k^i and d_l^i corresponding to the two nodes is $S_1^{'}(d_k^i,d_l^i)$, and their direct interaction intensity is $\tau S_1^{'}(d_k^i,d_l^i)+(1-\tau)S^{''}(d_k^i,d_l^i)$, denoted as $DS_i(d_k^i,d_l^i)$. Therefore, the sum of the direct interaction intensity between the developers in X_i is $\sum_{l=1}^{|X_i|}\sum_{k=1,k>l}^{|X_i|} DS_i(d_k^i,d_l^i)$, denoted as DS_i. The sum of the indirect cooperation intensity between developers in X_i is denoted as IS_i, and DS_i+IS_i is equal to $\sum_{l=1}^{|X_i|}\sum_{k=1,k>l}^{|X_i|} S(d_k^i,d_l^i)$. If $|X_i|$ is equal to 1, there will be only one developer in the community. As a result, the values of DS_i and DS_i+IS_i are both equal to 0.

The developer cooperation intensity in community is denoted as $f_1(X)$, which can be expressed as follows.

$$f_1(X)=\sum_{i=1,|X_i|\neq 1}^{I}\frac{DS_i}{DS_i+IS_i}\cdot\left(\frac{2\times(DS_i+IS_i)}{|X_i|}\right)^2 \tag{8}$$

2) Developer programming language similarity in community

If $|X_i|$ is larger than 1, we will take nodes v_k^i and v_l^i in X_i into consideration. The flag reflecting whether their corresponding developers master common programming language(s) is denoted as $Sim(d_k^i,d_l^i)$. If they master common programming language(s), one will have $Sim(d_k^i,d_l^i)=1$; otherwise, $Sim(d_k^i,d_l^i)=0$ will be held. Then, for all nodes in X_i, the developer programming language similarity in community can be calculated by $\frac{1}{|X_i|\cdot(|X_i|-1)}\sum_{l=1}^{|X_i|}\sum_{k=1,k>l}^{|X_i|} Sim(d_k^i,d_l^i)$, denoted as Sim_i. If there is only one node in X_i, the developer programming language similarity in community will be equal to 1, namely, $Sim_i=1$.

For all communities, their programming language similarity is denoted as $f_2(X)$, which can be calculated as follows.

$$f_2(X)=\sum_{i=1}^{I} Sim_i \tag{9}$$

In this way, the optimization model for community detection in the developer cooperative network can be represented as follows.

$$\begin{cases}\max f(X)=(f_1(X),f_2(X))\\ \text{s.t.}\quad X=\{X_1,X_2,\ldots,X_I\}\\ \qquad X_i\subseteq 2^V\\ \qquad X_i\cap X_j=\emptyset,\ i,j=1,2,\ldots,I\end{cases} \tag{10}$$

V. Adopted Evolutionary Optimization Algorithm

We adopt NSGA-II [29] as a multi-objective optimization framework, and incorporate it with some existing evolutionary operators to form a community detection algorithm, denoted as NSGA-DNet. Fig.1 depicts the overall framework of NSGA-DNet.

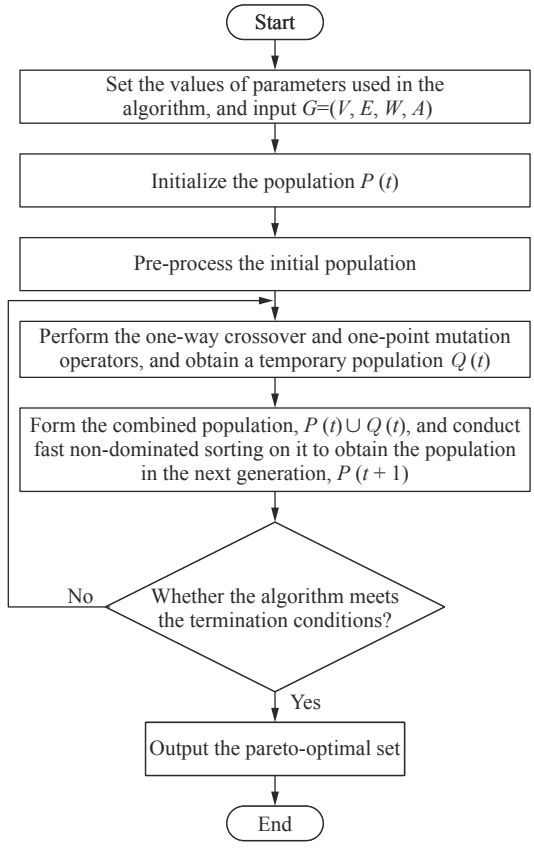

Fig. 1. Overall framework of NSGA-DNet.

In NSGA-DNet, the individual representation, population initialization, individual pre-processing, one-way crossover and one-point mutation operators are adopted according to [16], [19], [30] in this paper.

VI. Experiments

The following three groups of experiments are conducted in this section. The first one examines the influence of the weighted factor, τ, on community detection,

with the purpose of setting its appropriate value. The second one compares the method of calculating the developer cooperation intensity proposed in this paper with two existing methods, which attempts to evaluate the performance of our method. The third one compares the optimization model for the community detection established in this paper with others to show the rationality of our model. The experimental environment is Intel® Core™ i3-6100M CPU @ 3.70 GHz, 4G memory, and the programming language is Python.

1. Data acquisition

In order to verify the effectiveness of the method proposed in this paper, we choose GitHub, a popular code hosting platform, to collect data for experiments. First, a number of popular programming languages according to their ranks in GitHub are selected. Following that, repositories in each programming language are ranked in terms of the number of star signs, and a number of top repositories are selected. Next, a set of developers is formed by extracting contributors for each repository. Finally, a number of commit record sets of developers are formed by collecting all commit records of each developer from January 2015 to March 2020. If a repository contributed by a developer belongs to a programming language, the developer will master the programming language. All programming languages mastered by a developer constitute his/her programming language set. For a repository with a large number of developers, only 200 developers with the most contributions to the repository are reserved so as to have diverse data.

Four developer cooperation relationship and programming language based SECO networks are built, with their characteristics being listed in Table 1. In the table, NMVA represents the number of nodes, with the corresponding developers mastering multiple programming languages. Developers who commit to repositories belonging to different language are randomly chosen, so as to involve multiple languages in G_1 and G_4. In addition, G_4 expands a larger number of developers on the basis of G_1. If a developer does not have a direct cooperation relationship with all developers in a network, his/her corresponding node will be deleted from the network, so as to obtain a network without isolated nodes. The entire period is divided into two segments, namely, from January 2015 to January 2018 and from February 2018 to March 2020. Based on the two segments, the record set for each developer in G_4 is split to obtain G_2 and G_3. Compared with the other three networks, developers in G_4 cooperate more closely, and the characteristic in community gathering is more significant. Additionally, there are more developers who master multiple languages in G_4.

Table 1. Networks under test and their characteristics

Name	n	m	Average degree	Clustering coefficient	NMVA
G_1	1218	11638	19	0.6738	9
G_2	3066	35904	23	0.6625	89
G_3	2554	19401	15	0.6336	76
G_4	4248	67480	32	0.6844	114

2. Evaluation indicators and parameter settings

According to [9] and [31], the weighted modularity, Q, can reflect the significance of the community structure in the weighted network. The larger the Q value, the clearer the corresponding community structure. The entropy, EN, is utilized to measure the node attribute similarity in a community of a community division. The smaller the EN value, the more similar the corresponding node attribute similarity in a community of a community division. According to pre-experiments and previous studies [9], [31] for community detection in a network, the network is believed to have a community structure when the Q value is larger than or equal to 0.3, and the node attribute similarity in a community is acceptable when the EN value is smaller than or equal to 0.5. For a Pareto-optimal set, Q and EN are employed to evaluate each solution of the set, and their mean values are recorded, denoted as Q_{mean} and EN_{mean}, respectively. When comparing different Pareto-optimal sets, Q_{mean} and EN_{mean} will be adopted, whereas if different solutions are compared, Q and EN will be directly adopted. In addition, the average number of communities is counted in terms of the number of communities for each solution in the Pareto-optimal, denoted as ANC.

Suggested by [16] and [32], the parameter values of NSGA-DNet are set as follows: $\alpha = 0.2$, the population size, $PS = 100$, the crossover rate, $CR = 0.9$, the mutation factor, $MF = 0.1$, and the maximum number of generations, $T = 100$.

3. Influence of weighted factor on community detection

τ is set to 0, 0.1, 0.3, 0.5, 0.7, 0.9, and 1, respectively. Table 2 lists the Q_{mean} and EN_{mean} values obtained by NSGA-DNet under different values of τ, among which data in bold represent the best ones.

Table 2. Q_{mean} and EN_{mean} obtained by NSGA-DNet under different values of τ

Name	Indicator	$\tau=0$	$\tau=0.1$	$\tau=0.3$	$\tau=0.5$	$\tau=0.7$	$\tau=0.9$	$\tau=1$
G_1	Q_{mean}	0.4394	0.6492	0.648	**0.6723**	0.6711	0.6381	0.6579
	EN_{mean}	0.3595	**0.3284**	0.3403	0.3684	0.3716	0.3308	0.3403
G_2	Q_{mean}	0.2357	0.6107	0.6101	**0.6339**	0.6182	0.6231	0.6312
	EN_{mean}	0.4102	0.4469	0.4177	**0.3942**	0.4165	0.4496	0.4404
G_3	Q_{mean}	0.2694	0.6228	0.6476	**0.6641**	0.658	0.6531	0.6478
	EN_{mean}	**0.3251**	0.446	0.4267	0.4532	0.4248	0.4349	0.4398
G_4	Q_{mean}	0.2086	0.5777	0.586	**0.5926**	0.5885	0.5435	0.5917
	EN_{mean}	**0.3472**	0.3877	0.3789	0.361	0.3962	0.3654	0.4179

Table 2 shows that: 1) For all the networks, when $\tau=0.5$, the Q_{mean} value obtained by NSGA-DNet is the largest; 2) For G_2, G_3 and G_4, when $\tau=0$, the Q_{mean} value obtained by NSGA-DNet is smaller than 0.3, and all EN_{mean} values is smaller than 0.5; and 3) For the same values of τ, the Q_{mean} values of G_1, G_3, G_2 and G_4 gradually become smaller.

We can draw the following conclusion. The community structure obtained by NSGA-DNet is the most significant when the network topology and interaction behavior between developers are considered equally important in calculating the developer cooperation intensity. When only interaction behavior between developers is utilized to measure the developer cooperation intensity, the corresponding community division is generally unsatisfactory. As the network size increases, the significance of the community structure will decrease. All EN_{mean} values are acceptable, and developer programming languages in each community are similar. According to the experimental results under different values of τ for community detection, we will set the weighted factor, τ, to 0.5 in the subsequent experiments.

4. Influence of developer cooperation intensity on community detection

To illustrate the influence of different methods of calculating the developer cooperation intensity on community detection, the proposed method, denoted as IDCI, is compared with two existing methods. Since IDCI improves the method in [9] (denoted as DCI), it is necessary to compare IDCI and DCI. In addition, when the SECO network is unweighted [8], the default cooperation intensity between developers is equal to 1, and the corresponding method is denoted as EDCI. In order to have a fair comparison, the same parameter settings are adopted in NSGA-DNet when detecting communities based on IDCI, DCI, and EDCI, respectively. The experimental results are listed in Table 3.

Table 3. Q_{mean} and EN_{mean} obtained by NSGA-DNet under three methods of calculating the developer cooperation intensity

Name	Indicator	EDCI	DCI	IDCI
G_1	Q_{mean}	0.5406	**0.705**	0.6723
	EN_{mean}	0.3708	**0.3581**	0.3684
G_2	Q_{mean}	0.386	0.5744	**0.6339**
	EN_{mean}	0.4235	0.4092	**0.3942**
G_3	Q_{mean}	0.4519	0.6442	**0.6641**
	EN_{mean}	**0.4399**	0.4451	0.4532
G_4	Q_{mean}	0.3711	0.5814	**0.5926**
	EN_{mean}	0.3778	0.3814	**0.361**

From Table 3, we have the following observations: 1) For G_2, G_3, and G_4, the Q_{mean} value based on IDCI is larger than those based on its counterparts, and no matter which method of calculating the developer cooperation intensity is adopted, the Q_{mean} values of G_4 are the smallest; 2) For all the networks, the Q_{mean} value based on EDCI is the smallest; 3) For G_2 and G_4, the method of calculating the developer cooperation intensity based on IDCI is the best in terms of Q_{mean} and EN_{mean}.

The following conclusions can be drawn. For medium and large-scale networks, the proposed method of calculating the developer cooperation intensity has a positive influence on community detection. There are considerable differences of developer cooperation at different periods, which will lead to clear differences in different community divisions. If there are more developers and frequent interactions between them, the significance of the community structure will decrease.

Fig.2 shows a histogram of the ANC values based on EDCI, DCI and IDCI. From Fig.2, we have the following observations: 1) For G_1, G_2, and G_3, the ANC

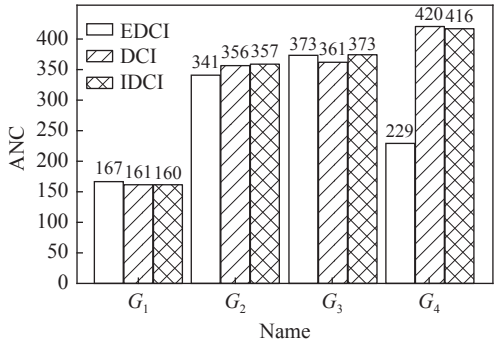

Fig. 2. Histogram of ANC based on EDCI, DCI, and IDCI.

values of different methods of calculating the developer cooperation intensity are very close, and for G_4, the ANC values of DCI and IDCI are much larger than that of EDCI; and 2) The ANC values of G_4 based on DCI and IDCI are larger than those of G_1, G_2, and G_3. For large-scale networks, the developer cooperation intensity significantly increases the number of communities, due to increasing the difference in cooperation between developers.

5. Comparison of optimization models

The multi-objective optimization model formulated in this paper is compared with two single-objective optimization models and five multi-objective optimization models, respectively, with the goal of illustrating the superiority of our multi-objective optimization model.

1) Comparison with two single-objective optimization models

We compare our model with two single-objective optimization ones to show the motivation of formulating a multi-objective optimization model. The two single-objective optimization models optimize the developer cooperation intensity in a community (see (8)) and the developer programming language similarity in a community (see (9)), respectively. For the fairness of comparison, GA-DNet and NSGA-DNet are used to solve single-objective and multi-objective optimization models, respectively, and their parameters are the same, with their experimental results being listed in Table 4. Due to multiple solutions obtained in each run by NSGA-DNet, we choose two solutions in the Pareto-optimal set, with one having a close Q value to that obtained by GA-DNet when optimizing (8) in the compared single-objective optimization model (columns 3 and 4 of Table 4). The other solution has a close EN value to that obtained by GA-DNet when optimizing (9) in the compared single-objective optimization model (columns 5 and 6 of Table 4).

Table 4. Q and EN of multi- and single-objective optimization models

Name	Indicator	Formula (8)	Formula (10)	Formula (9)	Formula (10)
G_1	Q	0.6863	0.6894	0.5843	0.6512
	EN	0.5274	0.4142	0.3163	0.3517
G_2	Q	0.6266	0.6267	0.5181	0.6093
	EN	0.5209	0.3801	0.3663	0.3641
G_3	Q	0.6922	0.6777	0.5611	0.639
	EN	0.5411	0.4745	0.3713	0.4129
G_4	Q	0.5922	0.5991	0.5492	0.5764
	EN	0.4653	0.3807	0.3768	0.3653

Table 4 tells that, 1) For G_1, G_2, and G_4, the multi-objective optimization model obtains better values of Q and EN than the single-objective optimization model formulated with (8). When the two models have close Q values, the multi-objective optimization model has smaller EN values than its counterpart; and 2) For all networks, the multi-objective optimization has the EN values of not completely smaller than the single-objective optimization model formulated with (9). However, its Q values are larger than those of its counterpart.

To sum up, the formulated multi-objective optimization model can not only provide various satisfactory community division schemes, but also achieve a balance between objectives.

2) Comparison with five multi-objective optimization models

Each of five community structure functions are first combined with the developer programming language similarity in a community to form the corresponding multi-objective optimization models. Then, the five multi-objective optimization models are compared with the proposed model.

Taking most previous optimization models are suitable for unweighted networks, we examine the community structure functions F1–F5 [14], [16], [17], [33] of the optimization model for unweighted networks, and extend them for the weighted SECO network, with their functions being denoted as FW1–FW5. Table 5 lists their mathematical expressions.

In Table 5, k_i^{in} stands for the number of edges in community X_i, k_i^{out} denotes the number of edges

Table 5. Mathematical expressions of F1–F5 and FW1–FW5

Term	Mathematical expression	Term	Mathematical expression
F1 [17]	$f_1 = \sum_{i=1}^{I} \frac{k_i^{out}}{2k_i^{in}+k_i^{out}}$	FW1	$fw_1 = \sum_{i=1}^{I} \frac{ODS_i}{2DS_i+ODS_i}$
F2 [14]	$f_2 = \sum_{i=1}^{I} (\sum_{l\in X_i} (\frac{k_i(l)}{\lvert X_i\rvert})^2) \times \frac{2k_i^{in}}{\lvert X_i\rvert}$	FW2	$fw_2 = \sum_{i=1}^{I} (\sum_{l\in X_i} (\frac{DS_i(l)}{\lvert X_i\rvert})^2) \times \frac{2DS_i}{\lvert X_i\rvert}$
F3 [33]	$f_3 = \sum_{i=1}^{I} \frac{k_i^{out}}{\lvert X_i\rvert}$	FW3	$fw_3 = \sum_{i=1}^{I} \frac{ODS_i}{\lvert X_i\rvert}$
F4 [16]	$f_4 = \sum_{i=1}^{I} \frac{k_i^{in}}{\lvert X_i\rvert} - \sum_{i=1}^{I} \frac{k_i^{out}}{\lvert X_i\rvert}$	FW4	$fw_4 = \sum_{i=1}^{I} \frac{DS_i}{\lvert X_i\rvert} - \sum_{i=1}^{I} \frac{ODS_i}{\lvert X_i\rvert}$
F5 [17]	$f_5 = \sum_{i=1}^{I} \frac{k_i^{out}}{2k_i^{in}} - \sum_{i=1}^{I} \frac{k_i^{out}}{2(m-k_i^{in})+k_i^{out}}$	FW5	$fw_5 = \sum_{i=1}^{I} \frac{ODS_i}{2DS_i} - \sum_{i=1}^{I} \frac{ODS_i}{2(W_t-DS_i)+ODS_i}$

between X_i and other communities, $k_i(l)$ represents the number of edges between node v_l^i in X_i and others in the same community, $DS_i(l)$ refers to the sum of the direct interaction intensities between developer d_l^i in X_i and others in the same community, ODS_i means the sum of the direct interaction intensities between developers in X_i and other communities, and the sum of the direct interaction intensities between developers in the network is denoted as W_t.

FW1–FW5 are combined with the developer programming language similarity in a community to form five corresponding multi-objective optimization models, denoted as FWA1–FWA5, respectively. For the five multi-objective optimization models and our model, NSGA-DNet is employed to detect communities in the built SECO networks, with the experimental results being listed in Table 6. In the table, data in bold represent the best Q_{mean} or EN_{mean}.

Table 6. Q_{mean} and EN_{mean} of six multi-objective optimization models

Name	Indicator	FWA1	FWA2	FWA3	FWA4	FWA5	Formula (10)
G_1	Q_{mean}	0.5692	0.6284	0.5294	0.6461	0.5556	**0.6723**
	EN_{mean}	0.3166	0.3306	**0.3098**	0.3745	0.3411	0.3684
G_2	Q_{mean}	0.5642	0.6151	0.5595	0.5982	0.5348	**0.6339**
	EN_{mean}	0.3957	0.4492	0.4034	0.3905	0.4127	**0.3942**
G_3	Q_{mean}	0.5441	0.6518	0.5688	0.6505	0.5483	**0.6641**
	EN_{mean}	**0.3637**	0.4289	0.4106	0.3904	0.382	0.4532
G_4	Q_{mean}	0.5595	0.5805	0.5048	0.5773	0.5292	**0.5926**
	EN_{mean}	0.4057	0.3987	0.3814	0.3705	0.3961	**0.361**

From Table 6 , 1) For all the networks, the proposed model is better than its counterparts in terms of Q_{mean}; 2) All Q_{mean} values are larger than 0.3, and all EN_{mean} values are smaller than 0.5; and 3) For G_2 and G_4, our model is the best in terms of Q_{mean} and EN_{mean}.

In summary, all models help to obtain the division schemes with the community structure and similar node attribute in a community. However, the developer cooperation intensity in a community is beneficial to obtain better community division.

VII. Conclusions and Future Work

This paper aims at solving the community detection problem of real SECO networks, which is beneficial to reveal intricate relationships between individuals. In view of this, we have built an SECO network by taking the developer cooperation relationship and programming language into consideration. Based on the network topology and interaction behavior between developers, we have proposed an accurate method of calculating the developer cooperation intensity, laying a foundation for community detection. From the two perspectives of the developer cooperation intensity and programming language similarity, we have formulated a multi-objective optimization model for community detection. Taking NSGA-II as the multi-objective evolutionary optimization framework, we have provided a community detection algorithm, NSGA-DNet, by incorporating it with several existing evolutionary operators. To evaluate the goodness of our work, four SECO networks have been built by crawling data in GitHub. The results demonstrate the effectiveness of our method and model for community detection in the SECO network.

It can be seen that SECO networks built in this paper are static, which has a difficulty in reflecting the evolution of developer cooperation at different times. As a result, the study on a dynamic SECO network is expected, especially the problem of community detection in the network. In addition, only one attribute is taken into consideration when building SECO networks. If we take more attributes of a node into consideration, the

built SECO networks will be more complex with more meaningful. How to address the issue resulted from more attributes of a node in SECO networks is another topic to be further studied.

References

[1] O. Franco-Bedoya, D. Ameller, D. Costal, *et al.*, "Open source software ecosystems: A Systematic mapping," *Information and Software Technology*, vol.91, pp.160–185, 2017.

[2] X. X. Zeng, W. Wang, C. Chen, *et al.*, "A consensus community-based particle swarm optimization for dynamic community detection," *IEEE Transactions on Cybernetics*, vol.50, no.6, pp.2502–2513, 2020.

[3] W. J. Luo, D. F. Zhang, H. Jiang, *et al.*, "Local community detection with the dynamic membership function," *IEEE Transactions on Fuzzy Systems*, vol.26, no.5, pp.3136–3150, 2018.

[4] J. Y. Chen, L. H. Chen, Y. X. Chen, *et al.*, "GA-based Q-attack on community detection," *IEEE Transactions on Computational Social Systems*, vol.6, no.3, pp.491–503, 2019.

[5] J. Chen, R. Li, S. Zhao, *et al.*, "A new clustering cover algorithm based on graph representation for community detection," *Acta Electronica Sinica*, vol.48, no.9, pp.1680–1687, 2020. (in Chinese)

[6] F. F. Wang, B. H. Zhang, and S. C. Chai, "Deep auto-encoded clustering algorithm for community detection in complex networks," *Chinese Journal of Electronics* , vol.28, no.3, pp.489–496, 2019.

[7] J. D. Fan, W. X. Xie, and Z. X. Liu, "A low complexity distributed multitarget detection and tracking algorithm," *Chinese Journal of Electronics*, in press, DOI: 10.23919/cje.2021.00.282, 2022.

[8] R. Bana and A. Arora, "Influence indexing of developers, repositories, technologies and programming languages on social coding community GitHub," in *Proceedings of the 11th International Conference on Contemporary Computing*, Noida, India, pp.1–6, 2018.

[9] T. T. Hou, X. J. Yao, and D. W. Gong, "Community detection in software ecosystem by comprehensively evaluating developer cooperation intensity," *Information and Software Technology*, vol.130, article no.106451, 2021.

[10] M. Girven and M.E.J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol.99, no.12, pp.7821–7826, 2002.

[11] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol.49, no.2, pp.291–307, 1970.

[12] M. E. J. Newman, " Fast algorithm for detecting community structure in networks," *Physical Review E*, vol.69, article no.066133, 2004.

[13] V. D. Blondel, J. L. Guillaume, R. Lambiotte, *et al.*, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol.2008, no.10, article no.P10008, 2008.

[14] C. Pizzuti, "GA-net: A genetic algorithm for community detection in social networks," in *Proceedings of International Conference on Parallel Problem Solving from Nature,* Dortmund, Rende, Italy, pp.1081–1090, 2008.

[15] L. L. Li, L. C. Jiao, J. Q. Zhao, *et al.*, "Quantum-behaved discrete multi-objective particle swarm optimization for complex network clustering," *Pattern Recognition* , vol.63, pp.1–14, 2017.

[16] M. G. Gong, B. Fu, L. C. Jiao, *et al.*, "Memetic algorithm for community detection in networks," *Physical Review E*, vol.84, article no.056101, 2011.

[17] F. Folino and C. Pizzuti, " An evolutionary multiobjective approach for community discovery in dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol.26, no.8, pp.1838–1852, 2014.

[18] S. Tahmasebi, P. Moradi, S. Ghodsi, *et al.*, "An ideal point based many-objective optimization for community detection of complex networks," *Information Sciences* , vol.502, pp.125–145, 2019.

[19] Z. T. Li, J. Liu, and K. Wu, "A multi-objective evolutionary algorithm based on structural and attribute similarities for community detection in attributed networks," *IEEE Transactions on Cybernetics*, vol.48, no.7, pp.1963–1976, 2018.

[20] A. Reihanian, M. Feizai-Derakhshi, and H. S. Aghdasi, "Community detection in social networks with node attributes based on multi-objective biogeography based optimization," *Engineering Application of Artificial Intelligence*, vol.62, pp.51–67, 2017.

[21] A. Idu, T. V. D. Zande, and S. Jansen, " Multi-homing in the Apple ecosystem: Why and how developers target multiple Apple app stores," in *Proceedings of International Conference on Management of Emergent Digital Ecosystems*, San Francisco, USA, pp.122–128, 2011.

[22] K. Plakidas, D. Schall, and U. Zdun, " Evolution of the R software ecosystem: Metrics, relationships, and their impact on qualities," *Journal of Systems and Software* , vol.132, pp.119–146, 2017.

[23] S. Jansen, " A focus area maturity model for software ecosystem governance," *Information and Software Technology*, vol.118, article no.106219, 2020.

[24] W. Ma, L. Chen, X. Y. Zhang, *et al.*, "How do developers fix cross-project correlated bugs? A case study on the GitHub scientific Python ecosystem," in *Proceedings of the 39th 2017 IEEE/ACM International Conference on Software Engineering*, Buenos Aires, Argentina, pp.382–392, 2017.

[25] Z. Y. Liu and Y. H. Ma, " A divide and agglomerate algorithm for community detection in social networks," *Information Sciences*, vol.482, pp.321–333, 2019.

[26] X. M. You, Y. H. Ma, and Z. Y. Liu, " A three-stage algorithm on community detection in social networks," *Knowledge-Based Systems*, vol.187, article no.104822, 2020.

[27] C. Pizzuti, " Evolutionary computation for community detection in networks: a review," *IEEE Transactions on Evolutionary Computation*, vol.22, no.3, pp.464–483, 2018.

[28] S. He, G. B. Jia, Z. X. Zhu, *et al.*, "Cooperative co-evolutionary module identification with application to cancer disease module discovery," *IEEE Transactions on Evolutionary Computation*, vol.20, no.6, pp.874–891, 2016.

[29] K. Deb, A. Pratap, S. Agarwal, *et al.*, " A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol.6, no.2, pp.182–197, 2002.

[30] D. Jin, J. Liu, B. Yang, *et al.*, "Genetic algorithm with local Search for community detection in large-scale complex networks," *Acta Automatica Sinica*, vol.37, no.7, pp.873–882, 2011. (in Chinese)

[31] X. Huang, H. Cheng, and J. X. Yu, "Dense community detection in multi-valued attributed networks," *Information Sciences*, vol.314, pp.77–99, 2015.

[32] X. Y Zhang, K. F. Zhou, H. B. Pan, *et al.*, "A network reduction-based multiobjective evolutionary algorithm for community detection in large-scale complex networks," *IEEE Transactions on Cybernetics*, vol.50, no.2, pp.703–716, 2020.

[33] B. A Attea and H. S. Khoder, "A new multi-objective evolutionary framework for community mining in dynamic social networks," *Swarm and Evolutionary Computation* , vol.31,

pp.90–109, 2016.

SHEN Xin received the M.S. degree from Jiangsu Normal University in 2019. He is a Ph.D. candidate of the School of Information and Control Engineering, China University of Mining and Technology. His research interests include mathematical modelling of complex systems, evolutionary optimization, and software ecosystem. (Email: shenxinpassion@163.com)

DU Junwei received the Ph.D. degree from Tongji University, in 2009. He is a Professor in the School of Information Science and Technology, Qingdao University of Science and Technology. His main research interests include software testing, natural language processing, knowledge graph and knowledge engineering. (Email: djwqd@163.com)

GONG Dunwei (corresponding author) received the Ph.D. degree from China University of Mining and Technology in 1999. He is a Professor in the School of Information and Control Engineering, China University of Mining and Technology. His current research interests include computation intelligence in many-objective optimization, dynamic and uncertain optimization, as well as their applications in software engineering, scheduling, path planning, big data processing and analysis. (Email: dwgong@vip.163.com)

YAO Xiangjuan received the Ph.D. degree from China University of Mining and Technology in 2011. She is a Professor in the School of Mathematics, China University of Mining and Technology. Her main research interests include intelligence optimization and software ecosystem. (Email:yaoxj@cumt.edu.cn)