PSO with Reverse Edge for Multi-Objective Software Module Clustering

Jiaze Sun\textsuperscript{a,b,*}, Yang Xu\textsuperscript{a}, and Shuyan Wang\textsuperscript{a,b}

\textsuperscript{*School of Computer Science and Technology, Xi’an University of Posts and Telecommunications, Xi’an, 710121, China}
\textsuperscript{Shaaxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi’an University of Posts and Telecommunications, Xi’an, 710121, China}

Abstract

The multi-objective software module clustering problem (MOSMCP) divides the complex software system into subsystems to obtain a perfect structure, which is based on the relations between modules to meet the conflicting software refactor objectives as much as possible. The modularization quality (MQ) and reverse edges number between clusters are considered as evaluation objectives, and a novel particle swarm optimization (PSO) with reverse edge, called REPSO, is proposed. First, the module dependency graph (MDG) in software system under clustering is constructed, and then the multi-objective particle swarm optimization (MOPSO) is improved to cluster the MDG. The exploring approach is used to update the particle locations. Four typical open source projects for module clustering are selected to verify the effectiveness of the REPSO. The laboratorial results prove that the REPSO algorithm is very effective and stable, and the diversity of the optimal solution is good. The REPSO algorithm provides an efficient engineering method for MOSMCP, which enhances the software structure and maintainability.

Keywords: multi-objective software module clustering; multi-objective particle swarm optimization; modularization quality; reverse edge number

(Submitted on July 21, 2018; Revised on August 22, 2018; Accepted on September 15, 2018)

© 2018 Totem Publisher, Inc. All rights reserved.

1. Introduction

With the rapidly increasing complexity of software systems, intelligibility and maintainability are becoming more and more crucial in the software lifecycle. The software module clustering problem (SMCP) has become a demanding problem in reverse software engineering [1]. Software module clustering (SMC) divides the software system into some subsystems to obtain a perfect structure, such that each module completes its designed objectives as independently as possible [2-3]. SMC contributes to enhance the software structure, increase the intelligibility and maintainability of software, and decrease maintenance costs [3].

Because the software system structure can be converted to a sophisticated graph structure, the SMCP can be converted into a graph partition problem [4]. Literature [1] and literature [5] adopted the software Module Dependency Graph (MDG) for SMCP and obtained reasonable resolutions. In literature [6], the SMCP was described as a directed graph clustering that maximizes cohesion and minimizes coupling. The SMCP is a classical NP-hard problem, and there is not an effective deterministic method to solve SMCP to the exact optimum in practice. The SMCP has been thought of as a classical search-based engineering optimization problem [7], where meta-heuristic optimization algorithms were verified more efficiently to find approximate reasonable solutions with a reasonable amount of time against deterministic approaches.

Mancoridis et al. [8] treated SMCP as an optimal engineering problem and developed an automatic SMC tool called Bunch with the climbing algorithm (HC) and genetic algorithm (GA). Literature [9] applied the genetic algorithm (GA) to SMC. Literature [2] adopted the particle swarm algorithm (PSO) to settle the SMCP, which achieved better results than GA. These above literatures all adopted the single objective method with modularization quality (MQ) as the fitness value to evaluate the clustering results. The MQ combines the module clustering requirements of the low coupling and high cohesion in a single objective fitness function.

* Corresponding author.
E-mail address: sunjaze@xupt.edu.cn
However, there are always many other factors aside from the traditional MQ, such as the number of reverse edges between different clusters, the uniformity of each module, and so on. The SMCP is a typical multi-objective optimization problem where several inconsistent objectives should be balanced and optimized. With the increase in the clustering objective numbers and conflict in multiple objectives, the SMCP becomes more and more difficult to work out. Therefore, the multi-objective software module clustering problem (MOSMCP) has become a focus problem in software engineering.

Literature [10] presented the SMCP as a multi-objective problem. Two different multi-objective solution algorithms, the equal-size cluster algorithm and the maximizing cluster algorithm, were proposed to consider the cluster number and module uniformity and make the clustering solutions more rational. Literature [9] proposed a hyper-heuristic GA for the MOSMCP, which is a speedy and efficient heuristic search technique that minimizes coupling and maximizes cohesion as the clustering objectives. Literature [11] proposed a multi-objective artificial bee colony (ABC) algorithm to work out the MOSMCP in accordance with higher cohesion, lower coupling, better modularization quality, and shorter reverse module distance.

Even though those meta-heuristic algorithms have successfully settled the MOSMCP, the other notable meta-heuristic algorithms (for example multi-objective particle swarm algorithm (MOPSO)) have not been explored. The application of PSO [5]2 in the single objective SMCP shows its effectiveness. Compared with other evolutionary algorithms, the PSO algorithm is characterized by simplicity, effectiveness, fast convergence, and great global search ability. Multi-objective particle swarm optimization [12] inherits the advantages of traditional PSO and reduces the possibility of falling into the local extremum for the multi-objective evolutionary algorithm.

To work out the MOSMCP, we exploited the number of reverse edges in the software MDG [13] as another clustering objective. A novel PSO algorithm is proposed to enhance the traditional algorithm, where the exploring approach is used to update the particle locations. The main contributions are as follows:

(1) This paper proposes a novel multi-objective meta-heuristic algorithm, namely particle swarm optimization (PSO) with reverse edge, called REPSO, for the MOSMCP. The REPSO algorithm is designed by combining the reverse edge number and MQ as the optimization objectives.

(2) This paper introduces the exploring strategy to directly update the particle location for multi-objective PSO algorithm, which decreases the calculations of the traditional particle location updating.

(3) This paper presents quantitative results that evaluate the effectiveness and diversity of the proposed multi-objective PSO with experiments of four benchmarks testing projects.

2. Software Module Dependency Graph

Software module clustering is based on the interactive relationship among the modules, which is often extracted as a software module dependency graph in software engineering. There are several software module concepts of different granularity such as statement, function, method, class, and package. The method is moderate and widely used granularity in most software systems. Consequently, it is regarded as the module in our research. The methods of software are the graph nodes, and the method call relations are the edges.

**Definition 1** Software MDG: The software MDG can be represented as graph \( G = (V, R) \). \( G \) is a directed graph. \( V = \{v_1, v_2, \ldots, v_n\} \) is the method vertices set. \( R = \{\langle v, w \rangle | P(v, w)(v, w \in V)\} \), and \( R \) is the edges set. \( \langle v, w \rangle \) represents a directed edge from vertex \( v \) to vertex \( w \). The predicate \( P(v, w) \) denotes that the method \( v \) calls the method \( w \) in the clustering software. The graph is directed and unweighted.

**Definition 2** Method call matrix: Given: \( n \), method number in MDG; \( F = \{f_1, f_2, \ldots, f_j, \ldots, f_n\} \), method set; \( e \), edge number between these methods. The method call relationships can be expressed as an \( n \times n \) matrix \( B = (b_{ij})_{n \times n} \). If the method \( f_i(1 \leq i \leq n) \) calls the method \( f_j(1 \leq j \leq n) \), \( b_{ij} = 1 \); otherwise, \( b_{ij} = 0 \). Therefore, the element number with the value of 1 is \( e \). Consequently, the software MDG is converted to method call matrix \( B \). In the software module clustering algorithms, the method call matrix \( B = (b_{ij})_{n \times n} \) saves the software structure, which represents the software method call relationships.
3. MOSMCP

3.1. Modularization Quality

The twin classical objectives of low coupling and high cohesion in the software module clustering have been associated into a unitary objective function, namely software modularization quality (MQ). The coupling between cluster-\(i\) and cluster-\(j\) is represented by \(e_{i,j}\) in Equation (1):

\[
e_{i,j} = \begin{cases} 
0, & \text{if } i = j \\
\frac{E_{i,j}}{2 \times N_i \times N_j}, & \text{if } i \neq j 
\end{cases}
\]  

(1)

The cohesion of cluster-\(i\) is represented by \(\mu_i\) in Equation (2):

\[
\mu_i = \frac{M_i}{N_i^2}
\]  

(2)

In Equation (1) and Equation (2), the variables \(i\) and \(j\) represent cluster-\(i\) and cluster-\(j\) respectively. \(N_i\) denotes the module number in cluster-\(i\). \(E_{i,j}\) represents the call relationships number between the cluster-\(i\) and the cluster-\(j\). \(M_i\) denotes the call relationships number between internal methods of cluster-\(i\).

As shown in Equation (3) and Equation (4), \(CF_i\) represents the modularity essential, and \(m\) denotes the cluster number. In the process of the SMC optimization, the coupling is minimized, and the cohesion is maximized.

\[
CF_i = \begin{cases} 
0, & \text{if } \mu_i = 0 \\
\frac{\mu_i}{\mu_i + \frac{1}{2} \sum_{j=1, j\neq i}^{m} (e_{i,j} + e_{j,i})}, & \text{otherwise}
\end{cases}
\]  

(3)

\[
MQ = \sum_{i=1}^{m} CF_i
\]  

(4)

3.2. Average Reverse Edge Number Between Clusters

Direction of call relationships between different methods should be as consistent as possible in software design, which will help build a relatively independent function block for system maintenance and management [13]. It is necessary to consider the unidirectional performance between the different clusters.

To evaluate the unidirectional performance of modularization quality in software module clustering, \(f_{\text{dir}}\) is designed to calculate the average number of reversed edges of clustering solution [1]. The edge directions of different clusters are more unidirectional when \(f_{\text{dir}}\) is smaller.
\[ f_{we} = \frac{1}{m+1} \sum_{i=1}^{m} \min \left( L(\bar{V}_i, V_i), L(V_i, \bar{V}_i) \right) \]  

(5)

In Equation (5), \( L(V_i, \bar{V}_i) \) is the inter-edge number between cluster-\( i \) and cluster-\( j \). \( L(\bar{V}_i, V_i) \) denotes the inter-edge number between cluster-\( i \) and the other clusters. \( m \) denotes the cluster number.

3.3. MOSMCP Model

To obtain the optimal software module clustering solutions that can accomplish the several clustering objectives as soon as possible, the MOSMCP technique is diffusely used in software reverse engineering. MOSMCP is formalized as follows [14]:

\[
\begin{align*}
\text{Min } y & = f(x) = [f_1(x), f_2(x), \ldots, f_n(x)], \quad n = 1, 2, \ldots, N \\
\text{S.t. } & (x | e_j(x) \leq 0, x \in S, j = 1, 2, \ldots, M) \\
\text{Where } & x = (x_1, x_2, \ldots, x_k)^T \in X \\
& y = (y_1, y_2, \ldots, y_k)^T \in Y \\
& x_{k_{\min}} \leq x_k \leq x_{k_{\max}}, \quad k = 1, 2, \ldots, K
\end{align*}
\]

(6)

Among Equation (6), \( K \) is the dimension of decision variable. \( N \) is the total number of objectives. \( M \) is the number of constraint conditions. \( X \) is the \( K \) dimension decision vector. \( Y \) is the objective function value vector. \( e_j(x) \) is the \( j \)th constraint condition. \( j \in [1, M] \). \( f_i(x) \) represents the \( i \)th objective function, \( i \in [1, N] \). \( X \) represents the decision space. \( Y \) represents the target space. \( S \) represents the feasible solution space.

In the multi-objective optimization problem [3], a set of non-nominated optimal solutions, called the Pareto optimal solution set, is always accomplished. We choose the MQ and the average reversed edge number as the objectives in the MOSMCP optimization. Suppose there exists two clustering solutions \( x_1, x_2 \). If at least one objective function meets \( f_i(x_1) < f_i(x_2) \) and at least another one objective function meets \( f_j(x_1) \geq f_j(x_2) \), \( i, j = 1, 2, \ldots, N \), then \( x_1, x_2 \) are non-dominated. Otherwise, if all the objective functions meet \( f_i(x_1) \leq f_i(x_2) \) and \( \exists \ i \in (1, 2, \ldots, N) \ f_j(x_1) < f_j(x_2), i, j = 1, 2, \ldots, N \), it means that the clustering solution \( x_1 \) dominates the clustering solution \( x_2 \).

4. PSO with Reverse Edge for MOSMCP

With consideration of software module clustering problems, we present a method of software module clustering based on minimize call directionality through research on the existing multi-objective particle swarm optimization algorithm.

4.1. Coding and Population Initialization

If it is clustered into \( m(m \ll n) \) subsystems, the software system can be denoted as the set \( C = \{c_1, c_2, \ldots, c_m\} \). Each clustering solution can be expressed as an \( m \times n \) matrix \( A = (a_{ij})_{mn} \), where \( a_{ij} \) is the element in the matrix. If the method \( f_j(1 \leq j \leq n) \) belongs to the cluster-\( i \) \( (1 \leq i \leq m) \), \( a_{ij} = 1 \). Otherwise, \( a_{ij} = 0 \). Consequently, the module clustering solution can be expressed as matrix \( A = (a_{ij})_{mn} \) in Equation (7):
The matrix $A(\alpha_{ij})_{m\times n}$ as a module clustering code is updated in the module clustering optimization process. Each particle is encoded by the module clustering matrix, which represents a feasible clustering solution.

According to the discrete characteristics of the SMCP, the method of population initialization is realized as follows: If the software system is divided into $m$ subsystems, we should find the first $m$ nonadjacent vertexes in the graph degree as the clustering initial centers. Other vertexes in MDG are divided into different clusters on the shortest paths length from each vertex to the centers of the cluster centers. The initialization method ensures a better initial solution to find the optimal solution quickly in particle swarm optimization.

### 4.2. Particle Location Update

The software module clustering problem is a classic discrete problem. We propose the exploring approach as the location update method to generate the next particle location. In the iterative process, the MQ and the average reversed edge number as two clustering fitness objectives are introduced to evaluate the particle location update. The average fitness probability of every particle is used to select the cluster and accelerate the convergence speed of the MOPSO algorithm. The objective function comparison considers pareto dominance relation when the locations of the particles are updating, and non-dominated solutions are saved to close to the pareto front in the multi-objective search space.

We assume that the location of the $n^{th}$ particle in the $t^{th}$ step iteration is expressed as Equation (8):

$$A_n^t = \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} \\ a'_{21} & a'_{22} & \cdots & a'_{2n} \\ \vdots & \vdots & & \vdots \\ a'_{m1} & a'_{m2} & \cdots & a'_{mn} \end{bmatrix}$$

If the location of the $n^{th}$ particle in the $(t+1)^{th}$ step iteration is $A_n^{t+1}$, and the module $f_j(1 \leq j \leq n)$ belongs to the cluster-$i$, $a'_{ij} = 1$, $a'_{i2j} = 0$, $a'_{i3j} = 0$, \ldots, $a'_{im1j} = 0$, $a'_{im2j} = 0$, \ldots, $a'_{imnj} = 0$, $(1 \leq i \leq m, 1 \leq j \leq n)$, matrix $A_n^t$ is as Equation (9):

$$A_n^t = \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} \\ a'_{21} & a'_{22} & \cdots & a'_{2n} \\ \vdots & \vdots & & \vdots \\ a'_{m1} & a'_{m2} & \cdots & a'_{mn} \end{bmatrix}$$

We construct $m$ matrices $A_n^t(1 \leq i \leq m)$ as above and calculate the MQ fitness $fitm_n$ and the average reversed edge number fitness $fitr_n$ of each location of matrix $A_n^t(1 \leq i \leq m)$. Therefore, we obtain the average probability value $p_n$ of each location $A_n^t$ according to the $fitm_n$ and $fitr_n$ as Equation (10):
\[ p_{ji} = \frac{\sum_{k} \text{fitm}_{ik} + \sum_{k} \text{fitr}_{ik}}{2} \]

(10)

The method \( f_j \) belongs to the cluster-\( i \) with the probability \( p_{ji} \). Each method of the \( w \)-th particle that clusters it belongs in the \( (r+1) \)th step iteration. Instead of using the velocity location update formula in basic particle swarm optimization, location update is used instead of speed update.

4.3. The External Archive Updating

When the multi-objective particle swarm optimization is used to solve the MOSMCP, the external archive is introduced to store the good particles in the previous swarm. When the new particle can dominate any of the particles within the external archive or is non-dominated to the contents of the archive, it can be added into the external archive. It is obvious that the dominated particle in the external archive must be deleted. When the number of the particles in the external archive exceeds the bounded scale, the redundant particle will be randomly moved from the archive to keep the size of the external archive bounded.

4.4. The Basic Steps of MOSMCP Using PSO

According to the above two optimization objectives, coding and initialization of particles, and location updating of particles, the whole MOPSO software module clustering algorithm is shown in Algorithm 1:

**Algorithm 1** MOPSO algorithm for MOSMCP

<table>
<thead>
<tr>
<th>Input:</th>
<th>the method call matrix; the size of the population; cluster number; the size of external archive; multi-objective functions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>module clustering optimal solution set.</td>
</tr>
<tr>
<td>Step1.</td>
<td>Initialize the population and external archive.</td>
</tr>
<tr>
<td>Step2.</td>
<td>for each particle ( f_j ) in the population do:</td>
</tr>
<tr>
<td>Step3.</td>
<td>update the location of the particle ( f_j ) based on the exploring approach</td>
</tr>
<tr>
<td>Step4.</td>
<td>update the external archive and calculate the fitness for each particle based on the MQ and the average RE</td>
</tr>
<tr>
<td>Step5.</td>
<td>If the max iteration number is not satisfied, then go to step2.</td>
</tr>
<tr>
<td>Step6.</td>
<td>Report the results of external archive, that is the Pareto optimal solution set</td>
</tr>
<tr>
<td>Step7.</td>
<td>End</td>
</tr>
</tbody>
</table>

5. Experiment Evaluation

The primary goal of empirical study is to assess the new REPSO algorithm for MOSMCP. We implement the algorithm described in Section 4 and measure its effectiveness and stability. We intend to investigate the following research questions in this paper:

RQ1: How does the different swarm size affect the pareto optimal solutions of the REPSO?

RQ2: How is the solution distribution of the REPSO under the different iteration times?

RQ3: How does REPSO perform for the MOSMCP in algorithm stability?

5.1. Experiment Set

In the experiments, we used four real open source software projects in Table 1, which had been widely used in software testing research. All experiments were performed on Windows 10 64-bit using an Intel (R) Core (TM) i3 CPU 3.60GHz processor, 6 GB of main memory, JDK1.8, and Eclipse 4.3.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>#Modules</th>
<th>#Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtunis</td>
<td>Turing operating system project for educational purposes</td>
<td>20</td>
<td>57</td>
</tr>
<tr>
<td>ispell</td>
<td>Spelling and typographical error correction software system</td>
<td>24</td>
<td>103</td>
</tr>
</tbody>
</table>

Table 1. Four open source software system information
PSO with Reverse Edge for Multi-Objective Software Module Clustering

<table>
<thead>
<tr>
<th>Software system to manage multiple revisions of files</th>
<th>REPSO algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>rcs</td>
<td>29</td>
</tr>
<tr>
<td>bison</td>
<td>37</td>
</tr>
</tbody>
</table>

5.2. Experiments and Results

5.2.1. Solution Distribution for Different Swarm Sizes

When the iteration number of the REPSO algorithm is 100, the distribution of the non-dominated solution set for four test projects with different swarm sizes such as 10, 20, 50, 80, 100, and 150 is shown in Figure 1. The horizontal axis denotes the Reserve Edge (RE) number of the software system. The vertical axis indicates the Modularization Quality (MQ) of the software system. Every point in the picture represents an optimal non-dominated solution. From Figure 1, we can see that, with the increase of particle swarm size, the solution points of the test projects shift to the upper left area of the coordinate system, which means that the non-dominated set will become better and better. Upgrading the swarm size can increase the diversity of the population and make it easy for the particles to produce more excellent new particles. Therefore, the increase in particle population size can help optimize the non-dominated optimal solution set to a certain degree.

Figure 1. Solution distribution under different swarm sizes

5.2.2. Solution Distribution for Different Iteration Times

To observe the optimal solution distribution of the REPSO algorithm under the different iteration times, the fitness values of MQ and RE in the first 200 iterations of the four test objects were recorded. The solution distribution under different iteration times is shown in Figure 2. In this figure, the particle swarm size is 100, and the number of iterations is 10, 30, 50, 100, 150, and 200. We can see that the non-dominated solution set gets better and better as the iteration times increase. Moreover, when the iteration times exceed 100 times, the non-dominated solution set no longer changes significantly, and the non-dominated solution tends towards the optimal set.

Figure 2. Solution distribution under different iteration times
5.2.3. Solution Stability in Different Experiments

Figure 3 is the distribution of non-dominated solutions in four independent experiments for the four open source software systems using the REPSO algorithm. In Figure 3, the particle swarm scale is 100, and the iteration number is also 100. It can be seen from Figure 3 that the results of four independent experiments are relatively concentrated, which indicates that the REPSO algorithm is very stable for MOSMCP.

6. Conclusion

Aiming at the MOSMCP, the exploring approach based on fitness probability is introduced into the traditional PSO location update, and the reverse edges number between clusters are considered as the second evaluation objectives to evaluate the clustering solution. Experimental results show that the REPSO algorithm for the MOSMCP is very effective and stable. In future work, we intend to use more test projects of different scales to analyse the advantages and disadvantages of different techniques for the MOSMCP. Furthermore, we intend to use more evaluation objectives to measure effectiveness of the software clustering solution.
Acknowledgement

This work was financially supported through grants from the Key Discipline Construction of General Institutions of Higher Learning from Shaanxi Province, the Industrial Research Project of Shaanxi Province (2018GY-014), and the Innovation Fund of Xi’an University of Posts and Telecommunications (CXJJ2017046).

References


Jiaze Sun graduated from the School of Information Science and Technology at Northwest University with a Ph.D. in 2015. He is currently working as an associate professor in the School of Computer Science and Technology at Xi’an University of Posts and Telecommunications (XUPT). His interests include swarm intelligence optimization algorithm, software testing, and data mining.

Yang Xu is a Master’s student in the School of Computer Science and Technology at Xi’an University of Posts and Telecommunications. Her research interests include swarm intelligence optimization algorithm and software testing.

Shuyan Wang graduated from the School of Information Science and Technology at Northwest University with a Ph.D. in 2006. She is currently working as a professor in the School of Computer Science and Technology at Xi’an University of Posts and Telecommunications. Her interests include data mining and software testing.