A New Genetic Algorithm Tool for Clustering-Based Aspect Mining Using Static Analysis and Vector-Space Model

Mohammad Mhawish¹, Manjari Gupta²
¹(Department of computer science, Banaras Hindu University, India)
²(Department of computer science, Banaras Hindu University, India)

ABSTRACT
In this paper, we proposed a new genetic clustering algorithm based aspect mining. Aspect mining tries to identify crosscutting concerns in a software system. Clustering in aspect mining deals with grouping of methods in a software system where each group represents a crosscutting concern. Our approach stands on adjusting a standard genetic algorithm to solve clustering problems, which employed the genetic algorithm operators with suitable probability values. We used two vector space models to associate method and its attribute values. The distance between two methods is expressed using Euclidian distance. In our approach to calculate methods attribute values, we used open source library ASM [14], to analyze compiled java classes.

Keywords–Clustering, Genetic algorithm, Aspect mining, crosscutting Concern, Static analysis

1. INTRODUCTION
Aspect-oriented programming is a new prototype that is initiated to design and implementing crosscutting concerns to tackle the modularity limitation of object-oriented programming. A crosscutting concern is referred to as characteristic of a software system that is spread all over the system. With tangled implementation, an example for crosscutting concerns in a software system is logging, pooling of connection and security. The benefit of using AOP is to an elevated level of modularization in a source code. This modularization enhances software system maintenance and higher productivity.

Aspect mining is the trend of research that tries to identify crosscutting concerns in existing developed software system without using AOP, to refactor crosscutting concerns into aspect to achieve the aim of aspect-oriented programming. Clustering is a technique used in pattern classification aimed to group objects into classes or clusters. Objects or methods are grouped based on a specific criterion. Clustering is considered as unsupervised learning used in data mining for grouping a similar data in clusters. In clustering based aspect mining, which purpose is to group methods in software system to find crosscutting concerns, the methods clustering is done by attributes values that are calculated using static analysis techniques. A result of clustering is a set of clusters; each cluster contains methods that are closely related to one another than methods assigned to different clusters.

2. RELATED WORK
Several aspect mining techniques have been proposed that used clustering to identify crosscutting concerns. Marin et al. [10] have proposed an aspect mining technique that uses the fan-in metric. Their idea is to search for crosscutting concerns among the methods that have the value of the fan-in metric greater than a given threshold [3]. (Shepherd and Pollock 2005) used cluster analysis to perform aspect mining, using methods name as a measure of similarity; if two methods same name then they belong to the same cluster [12].

(He, Zhang & Hu 2005) used a clustering to cluster methods within a software application, the similarity of methods clustered based on whether there was a methods call from within other methods. (Gabriela & Moldovan 2006 &2010)[3, 12] proposed a new k-mean & HAC based clustering algorithm, used attributes values for each method to calculate a similarity between methods by Euclidian distance.

3. IDENTIFICATION CROSSCUTTING CONCERNS (CCC) IN OUR APPROACH
We proposed the steps for identifying CCC in software system from compiled java classes that have been taken as an input to produce clustered methods as shown in Fig.1.
The following steps illustrate the approach:

3.1. Static analysis
We have taken compiled java classes as an input. The advantage of working on a compiled code is that it becomes possible to analyze classes at run-time just before loaded into JVM. In our implementation, we used ASM library to analysis compiled java classes represented as byte code arrays by processing all class files provided java archive (jar file), and inspecting all methods, and considering a method invocation. We have filtered all methods belonging to build-in classes’ ex. String, String Builder, and the trivial methods getter and setter.

3.2. Calculation methods attribute value
Methods information that is collected in analysis step for calculating the attribute values for the methods is (FAN-IN, CC, and Bi), where FIV is the value of FAN-IN metric and CC is the number of calling classes. It is discussed in details in section 4.

3.3. Convert to vector
To represent the attribute values for each method, we considered each method being characterized by L-Dimensional vector space model \( m = \{m_1, m_2, \ldots, m_L\} \).

3.4. Applying Genetic clustering algorithm
The created vectors (that hold methods with its attribute values), are used as input for the genetic clustering algorithm to group the similar methods in each cluster. The output after clustering is a set of clusters, each cluster contains similar methods. These clusters are ranked based on their attribute values (FAN-IN or CC). The higher rank represents a higher possibility of aspect candidates.

4. PERFORM GENETIC ALGORITHM IN OUR APPROACH
Genetic algorithms are naturally inspired search and optimization method, was introduced by John Holland. They start by initializing chromosomes that are selected randomly from solution space; each chromosome has an associated fitness value which quantifies its value as a solution to the problem. The best chromosome determines that higher fitness value, selected to employ the genetic algorithm operator (crossover and mutation). To applying and adjusting standard genetic algorithm to solve a clustering based aspect mining, one need to determine the following:

- Initial number of clusters.
- Representation of a chromosome.
- Genetic algorithm operators to employ.
- Suitable parameter value for (population size, probability values for applying operators).

In this paper, we stand on the original genetic algorithm that was proposed by Holland to adjust this algorithm to solve clustering based aspect mining.

Algorithm 1: original genetic algorithm

| a pseudo code for a standard genetic algorithm: |
| Randomly generate a population of m parents |
| While Good Solutions not Found |
| Compute the fitness of each member of the current population |
| Define selection probabilities for each parent |
| Generate m offspring by probabilistically selecting parents to produce offspring |
| Select only the offspring to survive |
| End While |

Based on the approach of SERBAN & MOLDOVAN [12] we have adjusted and implemented a new genetic algorithm, shown below, for solving clustering based aspect mining.

Algorithm 2: our genetic clustering algorithm

www.ijsret.org
The following steps are describing details of the basic steps of our genetic clustering algorithm:

4.1. Determine the number of clusters
In other clustering algorithms there is needed to specify the number of clusters in advance manually (by user), in our implementation we used specific heuristic for identifying the number of clusters that will represent in each chromosome (individual) using following steps:

1. A quantity of methods from the system is the initial number of clusters p.
2. The first centroid selected as the farthest from all methods (the method that have the maximum sum of the Euclidean distance from other methods).
3. For the methods that were not selected as centroid, compute the minimum distance (dmin) from the next centroid chosen as the method (m) whose maximum of minimum distance (maxdmin) if this value is higher than a positive minimum distance: threshold (we suppose if dmin less than this threshold it already belongs to the same crosscutting concern (cluster)
4. The step 3 is repeated until all methods are either become a new centroid or merged into already selected centroid. In this way, the number of clusters will be decreased.

4.2. Chromosome representation
Depending on the definition of clustering (divides the data into groups of similar objects) the methods in the software system are the objects to be clustered, X={m1, m2, …,mn}, each method from X belongs to application classes in the software system.

Each method is represented by l-dimensional vector based on the vector space model [8]. In our approach, we have considered each method by two different models for the vector representation [8]:

1. The vector associated with the method m is \{FIV, CC\}, where FIV is the value of FAN-IN metric, CC is the number of calling classes. This model is denoted by M1
2. The vector associated with the method m is \{FIV, B1, B2, …,Bl-1\}, where FIV is the value of FAN-IN metric, BI is the vector corresponding to the application classes ACi (1 ≤ i ≤ l − 1) as follow:

   B_i=1 if m is called from at least one method belonging to application class ACi, otherwise B_i=0. This model is denoted as M2.

To cluster methods according to their similarity, one must define a measure of how close two methods are, or how well their values compare. A small distance between the methods should indicate high similarity, thus distance measure can be used to quantify dissimilarity.

Several distance measures are employed for clustering, the most commonly used is the Euclidean distance. To find the distance between mi and mj is expressed as [8]:

\[ dE(mi,mj) = \sqrt{\sum_{k=1}^{i} (mik − mjk)^2} \]

4.3. Initialize population
The next step in the genetic clustering algorithm is initialized population of chromosomes with regard each chromosome i is fixed length represented set of all methods in the software system in some particular order.

The first population chromosomes are selected randomly from the solution space, with randomly selected the centroid that is placed in each chromosome.

4.4. Fitness computation and selection
Genetic algorithm is used to find approximate solution to optimization and search problem. Genetic clustering algorithm based aspect mining can be described as a standard genetic algorithm which reduces the square sum error (SSE) of cluster dispersion.

To select the better chromosomes that are passed to the next generation, the measure of cluster variance (SSE) play the role of the fitness function:
\[ SSE(K) = \sum_{j=1}^{p} \sum_{i=1}^{nj} d^2(m^j_i, f^j) \]

Where \( K = \{K_1, K_2, ..., K_p \} \) is referred to a partition of particular system. \( K_j \) is referred to a set of methods \{mj_1, mj_2, ..., mj_{nj} \} and \( f^j \) is a centroid of \( K_j \).

Thus \( K \) is the partition represented by chromosome \( i \), the fitness function of chromosome \( i \) is \( \text{fitness} = \text{MAX-SSE}(K) \), where \( \text{MAX} \) represent the maximum value of the square sum error (square sum of the distance between method \( m \) and the far centroid), \( \text{SSE} \): square sum error (square sum of the distance between method \( m \) and the nearest centroid).

The fitness function performed quantifies the suitability of each chromosome and used as a basis for selecting chromosomes for reproduction.

To perform clustering through genetic algorithm the starting point is to initialize random population of chromosomes and then the step of applying genetic algorithm operators is repeated until the termination condition is reached.

The input parameter to the next generation is the best chromosomes from the current generation, with a fixed replacement percentage (10%).

**4.5. Crossover:**
Crossover is a probabilistic process that exchanges information between two selected parents to generate two offspring [13]. In our approach, we used Roulette wheel method to select two parents from the current generation.

In our approach, we applied one-point crossover with fixed crossover probability (0.8) that is used by selecting a crossover point randomly within the chromosome string.

**4.6. Mutation**
Offspring are allowed to undergo swap mutation, conducted by randomly selected two centroids with randomly selected dimension and swapping between selected dimensions. Finally, we placed the offspring in the next generation. When a particular number of generations are reached, the chromosome with a maximum fitness function value corresponds to the optimal partition of the software system. This chromosome is the optimal solution represented a new number of clusters because the number of clusters will decrease during execution of the algorithm (by eliminating the empty clusters) and thus the number of clusters will be decreased.

**5. EVALUATION**

**5.1. Quality measure**
In order to evaluate the result obtained by our implementation of genetic algorithm, the evaluation is made from two points of view:

1. Succeed algorithm in discovered crosscutting concern
2. Succeed algorithm to distribute partitions and distribute methods in each cluster.

For this evaluation, we have used following quality measure [3, 4]:
- \( \text{PREC} \): the percentage of crosscutting concern are discovered
- Intra-cluster distance of a partition
- Inter-cluster distance of a partition

**5.1.1. PREC: the percentage of crosscutting concern are discovered.**
Let \( T \) be a clustering based aspect mining technique. The precision of a clustering based aspect mining technique \( T \) with respect to a partition \( K \) and the set CCC, denoted by \( \text{PREC}(\text{CCC}, K, T) \), is defined as:

\[ \text{PREC}(\text{CCC}, K, T) = \frac{1}{|\text{CCC}|} \sum_{i=1}^{|\text{CCC}|} \text{prec}(c_i, K, T). \]

Where
\[ \text{prec}(c_i, K, T) = \begin{cases} 1 & \text{if } c_i \text{ was discovered by } T \\ 0 & \text{otherwise} \end{cases} \]

Is the precision of \( T \) with respect to the crosscutting concern \( C_i \). \( \text{PREC} \) (CCC, K, T) defines the percentage of crosscutting concerns that are discovered by T. Only a part of the clusters are analyzed, meaning that some crosscutting concerns may be missed.

**5.1.2. Intra-cluster distance of a partition - IntraD.**
The intra-cluster distance of a partition \( K \), denoted by \( \text{IntraD}(K) \), is defined as:

\[ \text{IntraD}(K) = \sum_{j=1}^{p} \sum_{i=1}^{nj} d^2(m^j_i, f^j) \]

Where the cluster \( K_j \) is a set of methods \{mj_1, mj_2, ..., mj_{nj} \} and \( f^j \) is the centroid (mean) of \( K_j \). From the point of
view of a clustering technique, smaller values for IntraD indicate better partitions, meaning that IntraD has to be minimized.

5.1.3. Inter-cluster distance of a partition - InterD.
The inter-cluster distance of a partition $K$, denoted by $\text{InterD}(K)$, is defined as:

$$\text{InterD}(K) = \sum_{j=1}^{p} \sum_{i=1}^{p} d^2(f_i,f_j)$$

where $f_i$ and $f_j$ represent the centroids of clusters $K_i$ and $K_j$.
From the point of view of a clustering technique, greater values for InterD indicate better partitions, meaning that InterD has to be maximized.

5.2. Case study and result
For evaluation, we have used open source software “JHOTDRAW5.2” [11] that is a java framework for drawing 2D graphics. We used a set of crosscutting concerns to evaluate our algorithm: Consistent behaviour, Composite, Observer, Command, Contract Enforcement, Decorator, and Change Monitoring. The set of crosscutting concerns and their implementing methods were constructed using the results reported by Marin et al. in [10].
The comparative result between our genetic clustering algorithm and KAM algorithm described in [4], for both two models based vector space model, are presented in Table 1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PREC</th>
<th>$\text{InterD}$</th>
<th>$\text{IntraD}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GA using M1 model</strong></td>
<td>0.71</td>
<td>129.531</td>
<td>2206498.512</td>
</tr>
<tr>
<td><strong>GA using M2 model</strong></td>
<td>Out of time</td>
<td>Out of time</td>
<td>Out of time</td>
</tr>
<tr>
<td><strong>KAM using M1 model</strong></td>
<td>0.875</td>
<td>234.229</td>
<td>2263986.669</td>
</tr>
<tr>
<td><strong>KAM using M2 model</strong></td>
<td>0.875</td>
<td>127.819</td>
<td>35362681.902</td>
</tr>
</tbody>
</table>

By observing the results, we found the genetic algorithm results close to the result of KAM algorithm in the measured quality we used. Thus, one can say, with improved genetic algorithm we will get a better result.

6. DESCRIPTION OF OUR GENETIC ALGORITHM IMPLEMENTATION AND RESULT
In our genetic clustering algorithm tool, a compiled java classes or java archive file (JAR) have been taken as input, to calculate method attributes values. In our result view, we took a JHOTDAW as a case study. Results are generated in view of two models for method attributes values for each method in software system. With an option to filter the methods by fan-in value, these steps formed a static analysis and calculated attributes values as shown in fig 2.

![Fig 2. Result of static analysis](image)

In our tool to apply the genetic clustering algorithm, the methods attribute values are taken as input that we calculated in previous steps. Before executing the genetic clustering algorithm, we should determine a suitable parameter values that are engaged with genetic algorithm operator (population size, crossover probability, number of generation and threshold between clusters) as shown in fig 3.

![Fig 3. Result of the genetic clustering algorithm](image)
After the completion of all steps, the result will be a set of clusters. Each cluster contains a set of methods that we suppose belonging to the same crosscutting concern. From observation the result and ranked clusters by the methods; with highest fan-in value ranked highest. We took one cluster with highest rank to find the concerns containing methods, shown in Table 2.

<table>
<thead>
<tr>
<th>Cluster # 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH/ifa/draw/util/StorableOutput/writeInt --Fan-in: (57.0) CC: (18.0)</td>
</tr>
<tr>
<td>CH/ifa/draw/util/StorableInput/readInt --Fan-in: (54.0) CC: (16.0)</td>
</tr>
<tr>
<td>CH/ifa/draw/util/StorableOutput/writeStorable --Fan-in: (22.0) CC: (18.0)</td>
</tr>
<tr>
<td>CH/ifa/draw/framework/DrawingView/checkDamage --Fan-in: (23.0) CC: (22.0)</td>
</tr>
<tr>
<td>CH/ifa/draw/framework/DrawingView/clearSelection --Fan-in: (30.0) CC: (21.0)</td>
</tr>
<tr>
<td>CH/ifa/draw/framework/DrawingView/drawing --Fan-in: (58.0) CC: (29.0)</td>
</tr>
<tr>
<td>CH/ifa/draw/util/UndoableAdapter/undo --Fan-in: (22.0) CC: (22.0)</td>
</tr>
</tbody>
</table>

The Fan-in value of “undo” method is 22 and CC value is 22, that’s mean this method used in 22 application classes within JHOTDRAW. We observed also many methods contain the word “undo” most of them placed in one cluster while the remaining separated in clusters, with the high tangling of using “undo” concern in JHOTDRAW thus it is clear to cast it as a crosscutting concern.

The other crosscutting concerns we observed is persistence and resurrection of figures, which is performed by methods that are inherited from StorableInput and StorableOutput as we shown in the table, this methods is separated in more than 40 classes in JHOTDRAW.

7. CONCLUSION

This paper investigates how aspect mining can be done using a genetic clustering algorithm and how the genetic algorithm operators deal in aspect mining approach. We have proposed a new adjusting of genetic algorithm with its implementation in a standalone tool that identify a choice for threshold value (distmin) and Suitable parameter values for (population size, probability of applying operators). We have used heuristic for choosing the initial number of clusters, with decreasing the number of clusters during algorithm execution. In the future work, we will improve genetic algorithm operators for increasing the accuracy of the result and we will try to propose a new fitness function for achieving a better result.

REFERENCES


[14]. ASM open source library http://asm.ow2.org/