A New Genetic Clustering Based Approach in Aspect Mining

GABRIELA ŞERBAN
Babeş-Bolyai University
Department of Computer Science
1, M. Kogalniceanu Street, Cluj-Napoca
ROMANIA

GRIGORETA S. MOLDOVAN
Babeş-Bolyai University
Department of Computer Science
1, M. Kogalniceanu Street, Cluj-Napoca
ROMANIA

Abstract: Clustering is a division of data into groups of similar objects. *Aspect mining* is a process that tries to identify crosscutting concerns in existing software systems. The goal is to refactor the existing systems to use aspect oriented programming, in order to make them easier to maintain and to evolve. This paper aims at presenting a new *genetic* clustering based approach in aspect mining. Clustering is used in order to identify crosscutting concerns. We evaluate the obtained results from the aspect mining point of view based on two new quality measures. The proposed approach is compared with another clustering approach in aspect mining ([1]) and a case study is also reported.

Key–Words: Clustering, Genetic algorithms, Aspect mining.

1 Introduction

1.1 Clustering

Clustering is a division of data into groups of similar objects ([5]). From a machine learning perspective, clusters correspond to *hidden patterns*, the search for clusters is *unsupervised learning*.

Clustering can be considered the most important *unsupervised learning* problem. Unsupervised classification, or clustering, aims to differentiate groups (classes or clusters) inside a given set of objects, with respect to a set of relevant characteristics or attributes of the analyzed objects.

Each of these subsets (groups, clusters) consists in objects that are similar between themselves and dissimilar to objects of other groups.

Let $X = \{O_1, O_2, \ldots, O_n\}$ be the set of objects to be clustered. Using the vector-space model, each object is measured with respect to a set of l initial attributes $A_1, A_2, \ldots, A_l$ and is therefore described by a l-dimensional vector $O_i = (O_{i1}, \ldots, O_{il}), O_{ik} \in \Re, 1 \leq i \leq n, 1 \leq k \leq l$. Usually, the attributes associated to objects are standardized in order to ensure an equal weight to all of them ([5]).

The measure used for discriminating objects can be any *metric* or *semi-metric* function d. In our approach, we have used the *Euclidian distance*:

$$d(O_i, O_j) = d_E(O_i, O_j) = \sqrt{\sum_{k=1}^{l}(O_{ik} - O_{jk})^2}.$$

The *similarity* between two objects O_i and O_j is defined as

$$sim(O_i, O_j) = \frac{1}{d(O_i, O_j)}.$$

We have chosen *Euclidian distance* in our approach, because we have obtained better results, from the aspect mining point of view, than using other metrics (Manhattan, Hamming).

A large collection of clustering algorithms is available in the literature. [5], [6] and [7] contain comprehensive overviews of the existing techniques. Most clustering algorithms are based on two popular techniques known as *partitional* and *hierarchical* clustering.

1.2 Aspect Mining

The Aspect Oriented Programming (AOP) is a new programming paradigm that is used to design and implement *crosscutting concerns* [8]. A *crosscutting concern* is a feature of a software system that is spread all over the system, and whose implementation is tangled with other features' implementation. Logging, persistence, and connection pooling are well-known examples of crosscutting concerns. In order to design and implement a crosscutting concern, AOP introduces a new modularization unit called *aspect*. Some of the benefits that the use of AOP brings to software engineering are: better modularization, higher productivity, software systems that are easier to maintain and evolve.

Aspect mining is a relatively new research direction that tries to identify crosscutting concerns in

already developed software systems, without using AOP. The goal is to identify them and then to refactor them to aspects, to achieve a system that can be easily understood, maintained and modified.

Crosscutting concerns in non AO systems have two symptoms: *code scattering* and *code tangling*. *Code scattering* means that the code that implements a crosscutting concern is spread across the system, and *code tangling* means that the code that implements some concern is mixed with code from other (crosscutting) concerns.

1.3 Related Work

There are just a few aspect mining techniques proposed in the literature that use *clustering* in order to identify crosscutting concerns ([1], [2], [9], [11]).

In [2] a clustering approach in aspect mining based on *k-means* and *hierarchical agglomerative* clustering techniques is proposed. This approach is improved in [1], by defining a new *k-means* based clustering algorithm in aspect mining (*kAM*).

In this paper we present a new *genetic* clustering based approach in aspect mining and we propose two new quality measures for evaluating the obtained results from the aspect mining point of view.

The paper is structured as follows. Section 2 defines the problem of aspect mining as a clustering problem. A new *genetic* clustering algorithm in the context of aspect mining (*GAM*) is proposed in Section 3. An experimental evaluation of our approach, based on some quality measures, is presented in Section 4. The obtained results are compared with the ones obtained by applying *kAM* algorithm ([1]). Some conclusions and further work are outlined in Section 5.

2 Clustering approach in the context of aspect mining

2.1 Theoretical model

In this section we present the problem of identifying *crosscutting concerns* as a clustering problem.

Let $M = \{m_1, m_2, ..., m_n\}$ be the software system, where $m_i, 1 \leq i \leq n$ is a method of the system. We denote by n ($|M|$) the number of methods in the system.

We consider a crosscutting concern as a set of methods $C \subset M$, $C = \{c_1, c_2, ..., c_{cn}\}$, methods that implement this concern. The number of methods in the crosscutting concern C is $cn = |C|$. Let $CCC = \{C_1, C_2, ..., C_q\}$ be the set of all crosscutting concerns that exist in the system M. The number of crosscutting concerns in the system M is $q = |CCC|$.

Definition 1 ***Partition of a software system*** M**.** *The set* $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ *is called a* ***partition*** *of the system* $M = \{m_1, m_2, ..., m_n\}$ *iff* $1 \leq p \leq n$, $K_i \subseteq M, K_i \neq \emptyset, \forall i,\ \ 1 \leq i \leq p$, $M = \bigcup_{i=1}^{p} K_i$ *and* $K_i \cap K_j = \emptyset,\ \forall i, j,\ \ 1 \leq i, j \leq p, i \neq j$.

In the following we will refer to K_i as the i-th *cluster* of $\mathcal{K}$ and to $\mathcal{K}$ as a *set of clusters*.

In fact, the problem of aspect mining can be viewed as the problem of finding a partition $\mathcal{K}$ of the system M.

2.2 Identification of crosscutting concerns

The steps for identifying the crosscutting concerns that have the scattered code symptom, are:

- **Computation** - Computation of the set of methods in the selected source code, and computation of the attribute set values, for each method in the set.
- **Filtering** - Methods belonging to some data structures classes like *ArrayList*, *Vector* are eliminated. We also eliminate the methods belonging to some built-in classes like *String*, *StringBuffer*, *StringBuilder*, etc.
- **Grouping** - The remaining set of methods is grouped into clusters using a clustering algorithm (in our approach *GAM*, *kAM*). The clusters are sorted by the average distance from the point 0_l in descending order, where 0_l is the l dimensional vector with each component 0 (l is the number of attributes characterizing a method).
- **Analysis** - The clusters obtained are analyzed in order to discover which clusters contain methods belonging to crosscutting concerns. We analyze the clusters whose distance from 0_l point is greater than a given threshold.

3 Our approach

In this section we propose a new genetic clustering algorithm in aspect mining (*GAM*) that will be used in the **Grouping** step of the crosscutting concerns identification process (Subsection 2.2).

In our approach, the objects to be clustered are the methods from the software system, $X = \{m_1, m_2, ...m_n\}$. The methods belong to the application classes or are called from the application classes.

Based on the vector space model, we will consider each method being characterized by a *l*-dimensional vector: $m_i = (m_{i1}, \ldots, m_{il})$. In our approach we have considered that the vector associated with the method m is $\{FIV, B_1, B_2, ...B_{l-1}\}$, where FIV is the fan-in value ([10]) of m and B_i $(1 \leq i \leq l-1)$ is 1, if the method m is called from a method belonging to the application class C_i, and 0, otherwise.

In this context, the distance between two methods m_i and m_j is expressed using the *Euclidian distance*, as:

$$d_E(m_i, m_j) = \sqrt{\sum_{k=1}^{l}(m_{ik} - m_{jk})^2}$$

The clustering approach that we propose in this section is based on the use of Genetic Algorithms (GA) ([3], [4]) and attempts to minimize the within cluster variance.

Evolution has proven to be a very powerful mechanism in finding good solutions to difficult problems. One can look at the natural selection as an optimization method, which tries to produce adequate solutions to particular environments.

In this section we propose *GAM* algorithm (*Genetic clustering in Aspect Mining*).

GAM is a standard GA that minimizes the square sum error (*SSE*) of the cluster dispersion:

$$SSE(\mathcal{K}) = \sum_{j=1}^{p}\sum_{i=1}^{n_j} d^2(m_i^j, f_j)$$

where $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ is a **partition** of the system M, the cluster K_j is a set of methods $\{m_1^j, m_2^j, ..., m_{n_j}^j\}$ and f_j is the centroid (mean) of K_j.

$$f_j = \left(\frac{\sum_{k=1}^{n_j} m_{k1}^j}{n_j}, \ldots, \frac{\sum_{k=1}^{n_j} m_{kl}^j}{n_j} \right)$$

GAM uses an heuristic for choosing the number *p* of clusters, heuristic that is particular to aspect mining:

(i) The initial number *p* of clusters is *n* (the number of methods from the system).

(ii) The method chosen as the first centroid is the most "distant" method from the set of all methods (the method that maximizes the sum of distances from all other methods).

(iii) For each remaining methods (that were not chosen as centroids), we compute the minimum distance ($dmin$) from the method and the already chosen centroids. The next centroid is chosen as the method m that maximizes $dmin$ and this distance is greater than a positive given threshold $distMin$. If such a method does not exist it means that m is very close to its nearest centroid nc and should not be chosen as a new centroid (from the aspect mining point of view, m and nc should belong to the same (crosscutting) concern). In this case, the number *p* of clusters will be decreased.

(iv) The step (iii) will be performed repeatedly, until *p* centroids will be reached.

We mention that at steps (ii) and (iii) the choice could be a non-deterministic one. In the current version of *GAM* algorithm, if such a non-deterministic case exists, the first selection is chosen. Improvements of *GAM* algorithm can tackle these kind of situations.

In our approach we have chosen the value 1 for the threshold $distMin$ (iii). The reason for this choice is the following: if the distance between two methods m_i and m_j is less or equal to 1, we consider that they are similar enough to be placed in the same (crosscutting) concern. We mention that, from the aspect mining point of view, using *Euclidian distance* as metric and the vector space model proposed above, the value 1 for $distMin$ makes the difference between a crosscutting concern and a non-crosscutting one.

We use a generational model as underlying mechanism for *GAM* implementation.

```
Initialize population P_0
For i:=1 to NoOfGenerations do
 Add the best individuals from P_{i-1} to P_i
 While P_i is not complete do
   Select two parents P1 and P2 from P_{i-1}
   Recombine the parents, obtaining two
        offspring O1 and O2
   Mutate each offspring
   Add the two offspring O1 and O2 to P_i
 EndWhile
EndFor
Select the best individual from P_{NoOfGenerations}
```

Regarding *GAM* algorithm we have to make the following remarks:

- Each individual i is a fixed-length (equal to the number of methods in the system, *n*) array of integer numbers and defines cluster membership, i.e., $i = (i_1, i_2, ..., i_n),\ 1 \leq i_j \leq p, \forall j, 1 \leq$

$j \leq n$, where *p* is the number of clusters obtained by the above presented heuristic and $i_j = r$ iff method m_j belongs to cluster K_r.

- The measure of within cluster variance (*SSE*) is used as a fitness function in *GAM*, meaning that if $\mathcal{K}$ is the partition generated by an individual i, the fitness function of i is: $fitness(i) = Max - SSE(\mathcal{K})$, where Max is the maximum value of the square sum error.

GAM starts by creating a random population of individuals.

The main idea of *GAM* algorithm is that the following steps are repeated until a given number of generations ($NoOfGenerations$) is reached:

- We put in the next generation the best b individuals from the current generation. b is an input parameter of GAM.
- Two parents are selected using a Monte Carlo selection procedure until we obtain a new complete generation.
- The parents are recombined (using one-cutting point crossover) in order to obtain two offspring.
- The offspring are considered for mutation, which is performed by replacing a randomly selected gene with a randomly generated value between 1 and p.
- Finally, we put the offspring in the next generation.

After the maximum number of generations is reached, the best individual from the obtained population is chosen, i.e, the individual with the maximum value for the fitness function. This individual will determine the optimal partition of the software system M.

4 Experimental Evaluation

In order to evaluate the results of *GAM* algorithm from the aspect mining point of view, we use two new quality measure (Subsection 4.1). These measures will be applied on a case study (Subsection 4.2). The obtained results will be reported in Subsection 4.3. Based on the obtained results, *GAM* algorithm will be compared with *kAM* algorithm proposed in [1].

The reason for this comparison is that the criterion used in *GAM* (the minimization of the square sum error), makes it similar to *k-means* algorithm. *kAM* is also a *k-means* based algorithm.

4.1 Quality Measures

In this subsection we present two new quality measures (*PAME* and *ACC*). These measures are used for evaluating the results of clustering based aspect mining techniques from the aspect mining point of view.

In the following, let us consider a partition $\mathcal{K} = \{K_1, \ldots, K_p\}$ of a software system M and $CCC = \{C_1, C_2, ..., C_q\}$ the set of all crosscutting concerns from M.

Such a partition can be obtained using a clustering algorithm, as *kAM* or *GAM*.

Definition 2 ***Accuracy of a clustering based aspect mining technique - ACC.***

Let $\mathcal{T}$ be a clustering based aspect mining technique.

The accuracy of $\mathcal{T}$ with respect to a partition $\mathcal{K}$ and the set CCC, denoted by $ACC(CCC, \mathcal{K}, \mathcal{T})$, is defined as:

$$ACC(CCC, \mathcal{K}, \mathcal{T}) = \frac{1}{q} \sum_{i=1}^{q} acc(C_i, \mathcal{K}, \mathcal{T}).$$

$$acc(C, \mathcal{K}, \mathcal{T}) = \begin{cases} \frac{|C \cap K_j|}{|C|} & \text{, if } K_j \text{ is the first cluster in which } C \text{ was discovered by } \mathcal{T} \\ 0 & \text{, otherwise} \end{cases}$$

is the accuracy of $\mathcal{T}$ with respect to the crosscutting concern C.

For a given crosscutting concern $C \in CCC$, $acc(C, \mathcal{K}, \mathcal{T})$ defines the proportion of methods from C that appear in the first cluster where C was discovered.

In all clustering based aspect mining techniques, only a part of the clusters are analyzed, meaning that some crosscutting concerns or parts of them may be missed.

Based on the above definition it can be proved that $ACC(CCC, \mathcal{K}, \mathcal{T}) \in [0, 1]$. For lack of space we will give only an informal proof.

$ACC(CCC, \mathcal{K}, \mathcal{T}) = 1$ iff $acc(C, \mathcal{K}, \mathcal{T}) = 1, \forall\ C \in CCC$. In all other situations, $ACC(CCC, \mathcal{K}, \mathcal{T}) < 1$.

Larger values for ACC indicate better partitions with respect to CCC, meaning that ACC has to be maximized.

Definition 3 ***Percentage of Analyzed MEthods for a partition - PAME.***

Let us consider that the partition $\mathcal{K}$ is analyzed in the following order: $K_1, K_2, ..., K_p$.

The percentage of analyzed methods for a partition K with respect to the set CCC, denoted by $PAME(CCC, \mathcal{K})$, is defined as:

$$PAME(CCC, \mathcal{K}) = \frac{1}{q} \sum_{i=1}^{q} pane(C_i, \mathcal{K}).$$

$pame(C, \mathcal{K})$ is the percentage of the methods that need to be analyzed in the partition $\mathcal{K}$ in order to discover the crosscutting concern C, and is defined as:

$$pame(C, \mathcal{K}) = \frac{1}{n} \sum_{j=1}^{r} |K_j|$$

where $r = max\{t \,|\, 1 \leq t \leq p$ and $K_t \cap C \neq \emptyset\}$ is the index of the last cluster in the partition $\mathcal{K}$ that contains methods from C.

$PAME(CCC, \mathcal{K})$ defines the percentage of the number of methods that need to be analyzed in the partition in order to discover all crosscutting concerns that are in the system M. We consider that a crosscutting concern was discovered when all the methods that implement it were analyzed.

Based on Definition 3, it can be proved that $PAME(CCC, \mathcal{K}) \in (0, 1]$.

Smaller values for $PAME$ indicate shorter time for analysis, meaning that $PAME$ has to be minimized.

4.2 Case Study

In order to evaluate the results of *GAM* algorithm, we consider as case study Carla Laffra's implementation of Dijkstra algorithm ([13]).

This case study is a Java applet that implements Dijkstra algorithm in order to determine the shortest path in a graph. It was developed by Carla Laffra and consists in **6** classes and **153** methods.

4.3 Comparative Analysis of the Results

In this subsection we present the results obtained after applying *GAM* algorithm described in Section 3 with respect to the quality measures described in Subsection 4.1, for the case study presented above. The obtained results are compared with the ones obtained after applying *kAM* algorithm ([1]). The comparison is made from the aspect mining point of view.

For *GAM* algorithm the population size used was 100, with a replacement percentage of 10%, a crossover probability of 0.9 and a mutation probability of 0.3, one-point crossover was applied and 10000 generations were evaluated.

The comparative results are presented in Table 1.

Case study	Algorithm	ACC	PAME
Laffra	*kAM*	0.667	0.200
Laffra	*GAM*	0.667	0.220

Table 1: Comparative results.

From Table 1 we observe that, from the aspect mining point of view, *kAM* algorithm provides better results. However, the results obtained by *GAM* are very close to those obtained by *kAM* and we intend to improved them (Section 5).

In our view, the vector space model used for clustering in aspect mining has a large influence on the obtained results from the aspect mining point of view. We can conclude that the vector space model used in the proposed clustering based aspect mining technique should be improved.

We are currently working on improving the vector space model used for clustering in aspect mining.

5 Conclusions and Future Work

We have presented in this paper a new *genetic* based clustering approach in aspect mining. We have proposed ***GAM*** algorithm, which is a standard GA that uses an heuristic for choosing the number of clusters, from the aspect mining point of view.

In order to evaluate the obtained results from the aspect mining point of view we have introduced two new quality measures: ACC and $PAME$.

We have also provided a comparison of the results obtained by applying *GAM* and *kAM* algorithm ([1]).

Further work can be done in the following directions:

- To use different representation schemes for *GAM* and to compare the qualities and shortcomings of the different representations.
- To determine mutation schemes that will increase the accuracy of the results.
- To use heuristics for initializing the population used in *GAM*.
- To identify the most suitable stopping criterion for *GAM*, instead of using a fixed number of generations.
- To use multicriterial clustering techniques in order to obtain better partitions ([15]).
- To identify a choice for the threshold $distMin$ that will lead to better results from the aspect mining point of view.

- To improve the vector-space model used in clustering. In our opinion, the vector space models have significantly influenced the quality of the results from the aspect mining point of view.
- To determine the metric used in clustering that will provide better results from the aspect mining point of view (Minkowski, etc).
- To apply this approach for other case studies like: JHotDraw ([14]), PetStore and TomCat, as in [10].

References:

[1] G. Serban and G.S. Moldovan, A new k-means based clustering algorithm in Aspect Mining, *Proceedings of the 8th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing* (SYNASC'06), Timisoara, Romania, 2006, to appear.

[2] G. S. Moldovan and G. Serban, Aspect Mining using a Vector-Space Model Based Clustering Approach, *Proceedings of Linking Aspect Technology and Evolution Workshop(LATE 2006)*, Bonn, Germany, 2006, to appear.

[3] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, Mass., 1996.

[4] D. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Boston, USA, 1989.

[5] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.

[6] A. Jain and R. Dubes, *Algorithms for Clustering Data*, Englewood Cliffs, New Jersey: Prentice Hall, 1998.

[7] A. Jain, M. N. Murty, and P. Flynn, Data clustering: A review, *ACM Computing Surveys*, vol. 31, no. 3, 1999, pp. 264–323.

[8] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin, Aspect-Oriented Programming, *Proceedings European Conference on Object-Oriented Programming*, Springer-Verlag, vol. 1241, 1997, pp. 220–242.

[9] L. He and H. Bai, Aspect Mining using Clustering and Association Rule Method, *International Journal of Computer Science and Network Security*, vol. 6, no. 2A, 2006, pp. 247–251.

[10] M. Marin, A. van, Deursen, and L. Moonen, Identifying Aspects Using Fan-in Analysis, *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004)*, IEEE Computer Society, 2004, pp. 132–141.

[11] D. Shepherd and L. Pollock, Interfaces, Aspects, and Views, *Proceedings of Linking Aspect Technology and Evolution Workshop (LATE 2005)*, 2005.

[12] G. S. Moldovan and G. Serban, Quality Measures for Evaluating the Results of Clustering Based Aspect Mining Techniques, *Proceedings of Towards Evaluation of Aspect Mining (TEAM) Workshop, ECOOP*, 2006, pp. 13-16.

[13] C. Laffra, Dijkstra's Shortest Path Algorithm, http://carbon.cudenver.edu/ hgreenbe/courses/ dijkstra/DijkstraApplet.html.

[14] JHotDraw Project, http://sourceforge.net/ projects/jhotdraw.

[15] J. Handl, J. Knowles, Multiobjective Clustering with Automatic Determination of the number of clusters, Technical report TR-COMPSYSBIO-2004-02 UMIST, Manchester, 2004.