Exploring The Use of Genetic Algorithm Clustering for Mobile App Categorisation

Afnan A. Al-Subaihin[1] and Federica Sarro[2]

[1] CCIS, King Saud University, Riyadh, Saudi Arabia
[2] University College London, London, United Kingdom
aalsubaihin@ksu.edu.sa, f.sarro@ucl.ac.uk

Abstract. Search-based approaches have been successfully used as clustering algorithms in several domains. However, little research has looked into their effectiveness for clustering tasks commonly faced in Software Engineering (SE). This short replication paper presents a preliminary investigation on the use of Genetic Algorithm (GA) to the problem of mobile application categorisation. Our results show the feasibility of GA-based clustering for this task, which we hope will foster new avenues for Search-Based Software Engineering (SBSE) research in this area.

Keywords: Software clustering · Mobile applications · Replication study.

1 Introduction

Automatic software categorisation is an ongoing research problem which aims to find similarity among software artefacts [4]. Such similarity can aid, for example, in detecting malicious software [8], requirements discovery [3][19] and mining similar software behaviour and analytics [2][14][18]. Mobile applications (apps) present a particularly interesting domain since the app store categorisation has been deemed unsuitable and alternative automated categorisations are advocated [2][8]. Furthermore, app stores boast various metadata that can be leveraged for this task [9,12].

Al-Subaihin et al. [1][2] have shown that categorising mobile apps according to their functionalities can provide a better categorisation than the current app store ones. However, their work focuses on comparing the effectiveness of various feature extraction techniques from textual corpora, and thus only uses one clustering algorithm: hierarchical clustering.

In this paper, we carry out a partial replication of the original study [1] to investigate whether the results can be improved using a Genetic Algorithm-based clustering algorithm, as evolutionary approaches were shown to be successful as clustering techniques in other application domains [10]. Specifically, we investigate four of the five research questions posed in the original study, but we shift the focus on the clustering approach rather than the feature extraction method. Firstly, as a sanity check, we measure the degree of difference between the two clustering solutions. Then we report the degree of improvement on the original app store clustering. We also report partial results of investigating the best value of K for the GA-clustering algorithm (GAC) and, finally, we report the efficiency of using the GA-clustering technique compared to hierarchical clustering. Our results reveal that using a GAC produces significantly better clustering of mobile applications than those observed in the app store categorisation, and those produced by hierarchical clustering. However, GAC fails to surpass the quality of the hierarchical clustering solution at higher K values. To the best of our knowledge, our study is the first to explore the viability of GA-based clustering solutions for mobile app categorisation, and, more in general, few studies have investigated search-based clustering in SE research (e.g., [5,6,11,15]). This study shows the viability of search-based clustering solutions for SE tasks, and we hope can open further avenues for SBSE research.

2 Replication Study Design

The original study extracted features from mobile apps descriptions using four different techniques and compared their effectiveness for clustering apps by using these features and only one clustering approach, i.e. hierarchical clustering [1]. The study found that extracting features using Latent Dirichlet Allocation (LDA) consistently performs well among the investigated feature extraction techniques. Therefore, this replication uses LDA as a feature extraction technique, investigates the effectiveness of GA as a clustering technique, and compares it to the LDA-based hierarchical clustering results as reported in the original study. In the following, we report further details on the empirical study design.

2.1 Research Questions

We investigate four of the five research questions from the original study.

RQ 0. How similar is the GA-based clustering solution to its hierarchical counterpart? Investigating how similar GAC results are to the hierarchical clustering solution is a sanity check before proceeding further in this study. If the results are identical or very similar, there is no value in investigating GAC further. As in the original study, the similarity is measured by using the Jaccard index, which is a commonly used measure for the agreement of two partitions. Jaccard index ranges from 0 (complete dissimilarity) to 1 (identical).

RQ 1. Can GAC improve on current app store categorisation and hierarchical clustering? This research question compares the quality of the GAC results to the original app store, which has 24 categories ($K = 24$). The answer will confirm whether the use of GAC can actually improve the status quo (app store categorisation) and the state-of-the-art (original study). As done in the original study, we measure the quality of the clustering solutions using the Silhouette score [17]. A Silhouette score is assigned to each data point (i.e. app) in the dataset based on its similarity to the apps in the same cluster and its dissimilarity to apps in other clusters. The Silhouette score of an entire clustering solution is the mean scores of all data points in the dataset, and it ranges from -1 (complete mis-assignments) to 1 (perfect assignments).

RQ 2. How does the choice of K affect the clustering quality of GAC? Selecting a suitable K (i.e. number of clusters) is an ongoing problem in cluster analysis. This RQ explores how much the choice of K affect the quality of the resulting clusters. Due to the large cost of running GAC, we initially test the quality of the randomly generated populations at the possible values of K. This is then further explored by running GAC over three different values of K (23, 98 and 397), and by comparing it to the Hierarchical clustering solution. As in RQ1, we use the Silhouette score to measure the clustering solution quality.

RQ 3. How efficient is GAC compared to Hierarchical clustering? It is well known that a GA can be costly to fine-tune and evolve. Therefore, it is important to report its efficiency in terms of run-time in order to properly weigh benefits over its costs.

2.2 Dataset

In order to answer these RQs, we used the same dataset as the original study [1][3]. This dataset contains 12,664 Android mobile applications belonging to 24 categories, which have been randomly sampled from the Google Play app store. A detailed description of how this data was collected can be found elsewhere [1].

2.3 GA Approach

In this study, we opted to use the GA clustering approach proposed by Maulik and Bandyopadhyay [13]. This algorithm was shown to be able to find a global optimum, and its variations are widely available as code libraries. In our experiment we used the GAMA R package (v. 1.0.3) [16], and modified it to enhance the initial population generation and the penalty function (see section 2.3 for the problem at hand[4]).

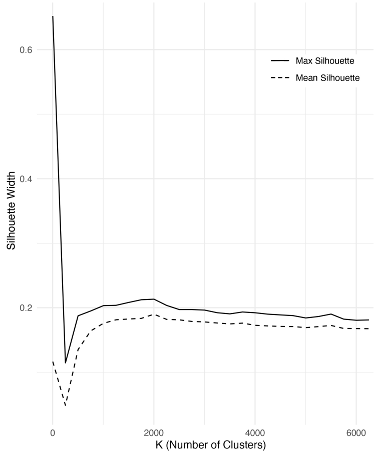

Fig. 1. **RQ 2**: Max (solid line) and mean (dashed line) silhouette scores (y-axis) of 500 random solutions at different values of K (x-axis) starting from k = 2 to k = dataset size / 2 and a step of 250.

Solution Representation and Evaluation The dataset of mobile apps is represented such that each mobile application is coded in terms of its LDA topics. The original study used 273 topics, thus, the dataset is a 12,664 by 273 matrix with each cell containing the relatedness of the app to the topic. Each individual is a clustering solution, which is encoded as a vector of real values, each representing a cluster centre [5] (with K known a priori). As the original study used the Silhouette score to measure the quality of a clustering solution, we have used this measure as fitness function for GAC. Upon generating the initial population of cluster centres randomly, they are evolved using linear rank selection, blend crossover and non-uniform mutation [13][21].

Empty Clusters Problem Upon generating the initial random population (i.e. random cluster centres), GAC looks at each gene and generates a uniformly random number between the upper and lower bounds found in the dataset. However, as our dataset represents topic relatedness (i.e. each gene in the individual is the relatedness of that individual to one specific topic), the dataset is a very sparse matrix, and the initial population of random centres can be very far from the actual data points in the dataset. As a result, many of the initial random solutions have mainly empty clusters as finding viable random cluster centres

[3] http://clapp.afnan.ws/data/

[4] Modified GAMA code can be found here: `https://github.com/afnan-s/gama`

[5] The cluster centre is the arithmetic mean of all the points belonging to the cluster.

that are sufficiently close to the data points is unlikely. In order to address this limitation, we have modified the random population generation such that it uniformly samples from the set of pre-observed values for each gene. In addition, we have adopted a penalty function that deducts the fitness of an individual proportionally to the number of empty clusters it contains.

Parameter Tuning and Setting We have explored running the GA with population sizes as low as 5, 10, 25, and 100. However, the cost significantly increases as the population size increases, since each individual consists of 273 (number of topics/genes) values multiplied by K (number of cluster centres). We have found that a good compromise is using population = 500 and generations = 1000, crossover rate = 0.9, mutation rate = 0.1. We also investigated decreasing crossover rate and increasing mutation rate, however, this did not produce better results.

Table 1. RQ 1: Silhouette width scores of existing app store categorisation, hierarchical clustering and GAC ($k = 24$, which is the number of categories in the app store).

	Min.	Max.	Mean	Median
Existing categorisation	-0.54	0.59	0.003	-0.01
Hierarchical Clustering	-0.64	0.99	0.02	-0.01
GAC	-0.49	1.00	0.52	0.55

3 Results and Discussion

This section presents the results for each of the investigated RQs, in addition to a discussion of the comparison to the original study and possible implications.

RQ 0. Similarity of Clustering Solutions When comparing the GAC clustering solution to the hierarchical clustering solution at K = 24 (with GAC having 1 empty cluster), the Jaccard similarity score is 0.37. This suggests that the solutions bear some similarity, however, as will be reported in following RQs, the GAC solutions are of significantly higher quality. This shows that GAC's solution does not stray much from the hierarchical clustering one, but indeed improves upon it.

RQ 1. Evaluation of Clustering Quality at Low K Table 1 shows summary statistics of the silhouette scores of each of the three clustering solutions. We observe that the GAC algorithm is able to produce significantly higher quality segmentation of the dataset at the same granularity of the app store. While in the original study, the hierarchical clustering solution improved upon the existing app store classification by 1.7%, the GAC solution improved it by 51.5% (note that 100% improvement means reaching a Silhouette score of 1). Indeed, the quality of the GAC-based solution exceeds that of the hierarchical-based one at its best selected K. We conclude that, upon requiring a coarser granularity clustering technique, GAC is a more suitable solution than a hierarchical technique, for this dataset.

Table 2. RQ 2: Mean Silhouette scores at three different levels of K using random cluster centres, hierarchical clustering, and GAC.

Categorisation Solution	K = 23	K = 98	K = 397
Random - Mean	-0.12	-0.05	0.13
Random - Best	0.13	0.05	0.19
Hierarchical Cluster.	0.03	0.13	0.3
GAC Cluster.	0.46	0.13	0.17

RQ 2. Best Overall K Upon studying the resulting clustering solutions of GAC for higher cluster numbers (i.e., higher K), the algorithm fails to produce

solutions with high mean silhouette scores at the given parameters, as shown in Table 2. This could provide evidence that, for the studied dataset, GAC may not be suitable for finer clustering granularity, though very competent at coarser ones. In order to gain further insight regarding the silhouette trend as K increases, we have investigated the mean silhouette score for random initial populations at different levels of K. The results (see Figure 1) suggest that silhouette scores are higher at lower K values and remain stagnant as K increases.

Table 3. **RQ 3**: Running time of the two clustering algorithms (measured on a standard laptop with an Intel Core i7 3.1 GHz and 16 GB RAM; d=days, h=hours, m= minutes, s=seconds).

No.	Steps	Hierarchical	GAC
1	Data Preprocessing	5.4 d	5.4 d
2	DTM Construction	3.0 h	3.0 h
3	Distance Matrix	21.0 s	1.3 m
4	Clustering	6.0 s	1.7 d

RQ 3. Efficiency Using LDA to represent the dataset of mobile app descriptions requires an upfront cost to search for the best LDA parameters that represent the data (further details can be found in [1]). Therefore, both GAC and Hierarchical clustering cost exactly the same for the first two steps (see Table 3). As GAC uses squared euclidean distance, as opposed to cosine distance used by the Hierarchical algorithm, it requires slightly longer to calculate the distance matrix. The major difference, however, can be observed in the time taken to produce the clustering solution. While Hierarchical clustering can produce a dendrogram in 6 seconds, from which a solution can be produced at any desired K (by cutting the denrogram), GAC can require several days on the same machine to produce a solution at any given K[6]. This shows that GAC might be too costly an option especially for larger K values, especially when considering the time taken to tune the parameters. However, due to the large improvement of the cluster quality over low K (from 0.02 produced by Hierarchical to 0.52 produced by GAC), this trade-off might be worthwhile. Moreover, the use of parallelisation when running GA can mitigate these costs [7] [20].

4 Conclusion and Future Work

This paper reports the initial results of our replication in which we investigate the efficacy of adopting a GA-based clustering approach to find a latent segmentation of mobile apps in the app store. We have found that GAC can be costly to run over a dataset comprising of a sparse matrix as typical of text analysis datasets. However, given a low enough K, GAC can exceed the results of hierarchical clustering with low enough cost given the improvement. On the other hand GAC did not produce clustering solutions over larger values of K that have higher quality than random search, possibly rendering it an unsuitable choice for finer granularity clustering. We plan to continue the line of investigation to fully replicate the original study and extend it. This includes investigating other similarity measurement techniques: vector space model, collocation- and dependency-based feature extraction methods as they might produce different results when combined with GAC. Also, we aim to further tune the GAC parameters to increase the confidence of the findings. Additionally, applying other search-based approaches may yield interesting results, including solving the empty

[6] Running time for GAC with k = 24, population = 500, generations = 1000.

cluster problem when generating random cluster centres by using a multi-objective GA that aims to maximise both the cluster quality and achieving different desired granularities (K). Our study sheds light on the feasibility of GA-based clustering for the SE task of mobile app categorisation. We hope this will foster further avenues for SBSE research in this area, as well as for many other clustering and classification tasks in SE [4].

References

1. Al-Subaihin, A., Sarro, F., Black, S., Capra, L.: Empirical comparison of text-based mobile apps similarity measurement techniques. EMSE **24**(6), 3290–3315
2. Al-Subaihin, A.A., Sarro, F., Black, S., Capra, L., Harman, M., Jia, Y., Zhang, Y.: Clustering mobile apps based on mined textual features. In: ESEM'16
3. AlSubaihin, A., Sarro, F., Black, S., Capra, L., Harman, M.: App store effects on software engineering practices. IEEE TSE (2019)
4. Auch, M., Weber, M., Mandl, P., Wolff, C.: Similarity-based analyses on software applications: A systematic literature review. JSS **168**
5. Ceccato, M., Falcarin, P., Cabutto, A., Frezghi, Y.W., Staicu, C.A.: Search Based Clustering for Protecting Software with Diversified Updates. In: SSBSE'16
6. Doval, D., Mancoridis, S., Mitchell, B.: Automatic clustering of software systems using a genetic algorithm. In: Procs. STEP '99. IEEE Comput. Soc
7. Ferrucci, F., Salza, P., Sarro, F.: Using Hadoop MapReduce for Parallel Genetic Algorithms: A Comparison of the Global, Grid and Island Models. ECJ **26**(4)
8. Gorla, A., Tavecchia, I., Gross, F., Zeller, A.: Checking app behavior against app descriptions. In: ICSE'14
9. Harman, M., Al-Subaihin, A., Jia, Y., Martin, W., Sarro, F., Zhang, Y.: Mobile app and app store analysis, testing and optimisation. In: MOBILESoft'16
10. Hruschka, E., Campello, R., Freitas, A., de Carvalho, A.: A Survey of Evolutionary Algorithms for Clustering. IEEE TCMCC **39**(2), 133–155 (2009)
11. Huang, J., Liu, J., Yao, X.: A multi-agent evolutionary algorithm for software module clustering problems. Soft Computing **21**(12) (2017)
12. Martin, W., Sarro, F., Jia, Y., Zhang, Y., Harman, M.: A survey of app store analysis for software engineering. IEEE TSE **43**(9), 817–847 (2017)
13. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern Recognition **33**(9), 1455–1465 (2000)
14. Nayebi, M., Farrahi, H., Lee, A., Cho, H., Ruhe, G.: More insight from being more focused: analysis of clustered market apps. In: WAMA'16
15. Praditwong, K., Harman, M., Yao, X.: Software Module Clustering as a Multi-Objective Search Problem. IEEE TSE **37**(2) (2011)
16. Rodrigues, J., Vasconcelos, G., Tin'os, R.: GAMA: Genetic Approach to Maximize Clustering Criterion (2019), `https://github.com/jairsonrodrigues/gama`
17. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. JCAM **20**, 53–65 (1987)
18. Sarro, F., Harman, M., Jia, Y., Zhang, Y.: Customer rating reactions can be predicted purely using app features. RE'18 (2018)
19. Sarro, F., Al-Subaihin, A.A., Harman, M., Jia, Y., Martin, W., Zhang, Y.: Feature lifecycles as they spread, migrate, remain, and die in app stores. In: RE'15
20. Sarro, F., Petrozziello, A., He, D.Q., Yoo, S.: A New Approach to Distribute MOEA Pareto Front Computation. In: GECCO'20
21. Scrucca, L.: GA: A package for genetic algorithms in R. Journal of Statistical Software **53**(4), 1–37 (2013)