A New Binary Similarity Measure Based on Integration of the Strengths of Existing Measures: Application to Software Clustering

Rashid Naseem(✉) and Mustafa Mat Deris

Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia
rnsqau@gmail.com

Abstract. Different binary similarity measures have been explored with different agglomerative hierarchical clustering approaches for software clustering, to make the software systems understandable and manageable. Similarity measures have strengths and weakness that results in improving and deteriorating clustering quality. Determine whether strengths of the similarity measures can be used to avoid their weaknesses for software clustering. This paper presents the strengths of some of the well known existing binary similarity measures. Using these strengths, this paper introduces an improved new binary similarity measure. A series of experiments, on five different test software systems, is presented to evaluate the effectiveness of our new binary similarity measure. The results indicate that our new measure show the combined strengths of the existing similarity measures by reducing the arbitrary decisions, increasing the number of clusters and thus improve the authoritativeness of the clustering.

Keywords: Binary similarity measures · Improved measure · Agglomerative hierarchical clustering · Software clustering

1 Introduction

Clustering is an approach that makes clusters of similar entities in the data. In the software domain, an important application of clustering is to modularize a software system or to recover the module architecture or components of the software systems by clustering the software entities, e.g. functions, files or classes, in the source code. Recovery is very important when no up-to-date documentation of a software system is available [1].

Agglomerative Hierarchical Clustering (AHC) algorithms have commonly used by researchers for software clustering [3,4]. AHC comprises of two main factors, a similarity measure to find the association between two entities and a linkage method to update the similarity values between entities in each iteration. However, selection of a similarity measure is an important factor in AHC [5], that has a major influence on the clustering results [7]. For software clustering the comparative studies has reported that Jaccard binary similarity measure produced better clustering results [7]. In our previous study [6], we proposed a new binary similarity measure, called JaccardNM, which could overcome some deficiencies of Jaccard binary similarity measure.

In this paper, we explore the integration of the existing binary similarity measures for AHC algorithms using linkage methods (e.g. Complete Linkage (CL) and Singel Linkage (SL)). For example, we select the Jaccard similarity measure, which produces a relatively large number of clusters [8,9] and the JaccardNM binary similarity measure which takes less number of arbitrary decisions [6,10]. Creating large number of clusters means that a clustering approach may creates compact clusters, hence improving the quality of clustering results [3]. Arbitrary decision is the arbitrary clustering of two entities when there exist more than two equally similar entities, hence arbitrary decisions create problems and reduce the quality of clustering results [3,6]. This analysis leads us to introduce better binary similarity measures by combining the Jaccard and JaccardNM measures, i.e. "Jaccam".

The paper is organized as follows: Sect. 2 illustrate the software clustering using AHC algorithm. Section 3 shows and analyze the strengths of the existing similarity measures and the new proposed similarity measure. Section 4 gives the experimental results and discussion on comparing our new similarity measure with existing similarity measures by using arbitrary decisions, number of clusters and authoritativeness as evaluation criteria. Section 5 concludes this paper.

2 Software Clustering Using AHC

Algorithm 1 presents the main steps of AHC, which starts by grouping the entities into small clusters in a bottom up fashion. In every iteration, AHC clusters the most similar entities until the targeted number of clusters is reached or a final large cluster that contains all entities is formed. When AHC is employed for the software clustering, the first step that occurs is the selection of the entities to be clustered where each entity is described by different features.

ALGORITHM 1. Agglomerative Hierarchical Clustering (AHC) Algorithm

```
Input: Feature (F) matrix
Output: Hierarchy of Clusters (Dendrogram)

initialization;
1 Create a similarity matrix by calculating similarity using a Similarity Measure between
each pair of entities;
repeat
    2 Group the most similar (singleton) clusters into one cluster (using maximum value of
    similarity in similarity matrix);
    3 Update the similarity matrix by recalculating similarity using a Linkage Method
    between newly formed cluster and existing (singleton) clusters;
until the required number of clusters or a single large cluster is formed;
```

2.1 Selection of Entities and Features

Selecting the entities and features associated with entities depends on the type of software system to be clustered. Researchers have used different types of entities

Table 1. An example feature ($E \times F$) matrix

	f1	f2	f3	f4	f5	f6	f7
E1	1	1	0	0	0	0	0
E2	1	1	0	0	0	0	0
E3	1	0	1	1	0	0	0
E4	0	0	1	1	1	0	0
E5	0	0	0	0	0	1	0

Table 2. The similarity matrix derived from the matrix in Table 1 by using the Jaccard similarity measure

	E1	E2	E3	E4	E5
E1					
E2	1				
E3	0.25	0.25			
E4	0	0	0.5		
E5	0	0	0	0.2	

e.g. files [11], classes [12] and methods [9]. Researchers have also used different types of features to describe the entities such as global variables used by an entity [2], procedure calls [11]. Features may be in binary or non-binary format. A binary feature represents the presence or absence of a feature, while non-binary features are weighted features, to demonstrate the strength of the relationship between entities. Binary features are widely used for software clustering [5,13].

When entities and features are extracted from a software system, it results in a feature matrix of size $E \times F$, where E is the total number of entities and F is the total number of features. AHC takes $E \times F$ as input, as shown in Algorithm 1. Table 1 shows an example 0–1 feature matrix *ExF*, which contains 5 entities (E1–E5) and 7 binary features (f1–f7). In Table 1, for example, f1 is present in entities E1, E2, and E3 while absent in entities E4 and E5.

2.2 Selection of Similarity Measure

The first step of the AHC process is to calculate the similarity between the entities to obtain a similarity matrix. For this purpose a similarity measure can be used. Some of the well known binary similarity measures:

$$Jaccard = a/(a+b+c) \tag{1}$$

$$JaccardNM = a/(2(a+b+c)+d) \tag{2}$$

All the existing binary similarity measures are expressed as combinations of the four quantities associated with the pair of entities (Ei, Ej): (1) the number of features common to both entities, denoted by a; (2) the number of features present in Ei, but not in Ej, denoted by b; (3) the number of features present in Ej, but not in Ei, denoted by c; (4) the number of features absent in both entities, denoted by d. It is important to note that a+b+c+d is equal to the total number of features F.

To illustrate the calculation of Jaccard measure as defined in Eq. 1, Table 2 gives the corresponding similarity matrix of the feature matrix shown in Table 1. The similarity between E1 and E2 is calculated using the quantities defined by a, b, c, and d, and in this case a = 2, b = 0, c = 0, and d = 5. Putting all these values in Jaccard similarity measure, we get similarity value ‘1’ (shown in Table 2). Likewise, similarity values are calculated for each pair of entities and are presented in Table 2. Now AHC will group the most similar entities in Table 2, according to the Step 2 in Algorithm 1. E1 and E2 have the highest similarity value, so AHC groups these entities in a single cluster (E1E2). A new cluster is therefore formed, and AHC will update the similarity values of E1E2 and all other (singleton) clusters, i.e., E3, E4, and E5. To update these similarity values different linkage methods can be used, which are described in the next subsection.

2.3 Selection of the Linkage Method

When a new cluster is formed, the similarities between new and the existing clusters are updated using a linkage method (Step 3 of Algorithm 1). There exist a number of linkage methods which update similarities differently. However, in this study we only discuss those linkage methods which are widely used for software clustering. They are listed below, where (EmEn) represents a new cluster and Eo represents an existing singleton cluster.

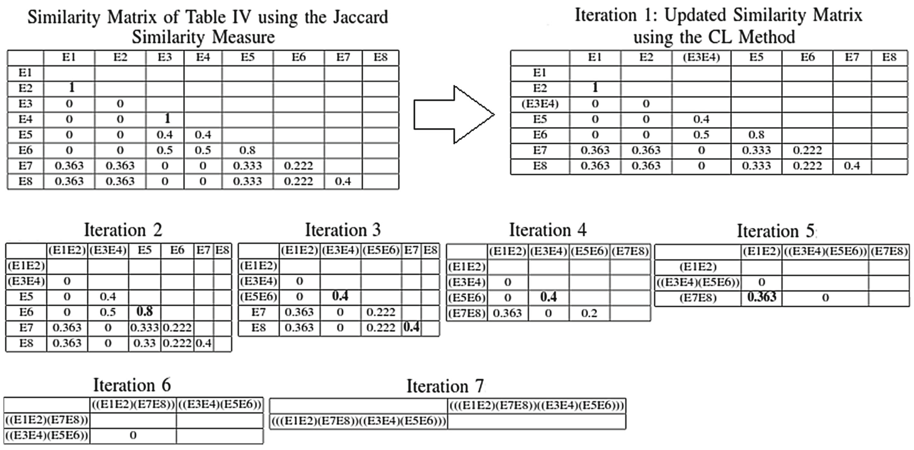
- *CL(EmEn, Eo)= min(similarity(Em, Eo), similarity(En, Eo))*
- *SL(EmEn, Eo)= max(similarity(Em, Eo), similarity(En, Eo))*

In the illustrative example, we update similarity values between a new cluster (E1E2) and existing singleton clusters using CL method. The updated similarity matrix is shown in Table 3. For example, the CL method returns the minimum similarity value between E1 and E3 (i.e., 0.25) and E2 and E3 (i.e., 0.25). Both of the returned values are the same (if there was a minimum, that would be selected), therefore, AHC selects this similarity value as the new similarity between (E1E2) and E3, as shown in Table 3. Similarly, all similarity values are updated between (E1E2) and E4 and E5.

Table 3. The updated similarity matrix from the values in Table 2 using CL linkage method

	E1E2	E3	E4	E5
E1E2				
E3	0.25			
E4	0	0.5		
E5	0	0	0.2	

AHC repeats Steps 2 and 3 until all entities are merged in one large cluster, or the desired number of clusters is obtained. At the end AHC results in a hierarchy of clusters, also known as dendrogram. The obtained hierarchy is then evaluated to assess the performance of similarity measures and linkage methods.



Similarity Matrix of Table IV using the Jaccard Similarity Measure

	E1	E2	E3	E4	E5	E6	E7	E8
E1								
E2	**1**							
E3	0	0						
E4	0	0	**1**					
E5	0	0	0.4	0.4				
E6	0	0	0.5	0.5	0.8			
E7	0.363	0.363	0	0	0.333	0.222		
E8	0.363	0.363	0	0	0.333	0.222	0.4	

Iteration 1: Updated Similarity Matrix using the CL Method

	E1	E2	(E3E4)	E5	E6	E7	E8
E1							
E2	**1**						
(E3E4)	0	0					
E5	0	0	0.4				
E6	0	0	0.5	0.8			
E7	0.363	0.363	0	0.333	0.222		
E8	0.363	0.363	0	0.333	0.222	0.4	

Iteration 2

	(E1E2)	(E3E4)	E5	E6	E7	E8
(E1E2)						
(E3E4)	0					
E5	0	0.4				
E6	0	0.5	**0.8**			
E7	0.363	0	0.333	0.222		
E8	0.363	0	0.33	0.222	0.4	

Iteration 3

	(E1E2)	(E3E4)	(E5E6)	E7	E8
(E1E2)					
(E3E4)	0				
(E5E6)	0	**0.4**			
E7	0.363	0	0.222		
E8	0.363	0	0.222	**0.4**	

Iteration 4

	(E1E2)	(E3E4)	(E5E6)	(E7E8)
(E1E2)				
(E3E4)	0			
(E5E6)	0	**0.4**		
(E7E8)	0.363	0	0.2	

Iteration 5

	(E1E2)	((E3E4)(E5E6))	(E7E8)
(E1E2)			
((E3E4)(E5E6))	0		
(E7E8)	**0.363**	0	

Iteration 6

	((E1E2)(E7E8))	((E3E4)(E5E6))
((E1E2)(E7E8))		
((E3E4)(E5E6))	0	

Iteration 7

	(((E1E2)(E7E8))((E3E4)(E5E6)))
(((E1E2)(E7E8))((E3E4)(E5E6)))	

Fig. 1. The similarity matrix and iterations in clustering process using Jaccard measure and CL method

3 The Jaccam Similarity Measure

As discussed in Sect. 1, we define a new similarity measure which has the combined strengths of the Jaccard and JaccardNM defined in Eqs. 1 and 2, respectively. To highlight the strengths of these existing measures we first show a small example case study, and then define our new similarity measure.

3.1 An Example Case Study

To illustrate the strengths of existing similarity measures, we take an example feature matrix (see Table 4). Feature matrix contains 8 entities (E1–E8) and 13 features (f1–f13). Using feature matrix shown in Table 4, we illustrate the strengths of Jaccard and JaccardNM similarity measures and CL method is used to updated the similarity matrix.

Jaccard with CL Clustering Process. First we illustrate the Jaccard measure with the CL method. The first step of AHC is to create the similarity matrix using a similarity measure. After applying Jaccard measure to feature matrix in Table 4, we get the similarity matrix shown in Fig. 1. In the first iteration of AHC, a maximum similarity value from the similarity matrix is selected to make a new cluster or to update a cluster. So, AHC searches for a maximum similarity value in the similarity matrix but it finds maximum similarity value '1' two times. Hence, there are two arbitrary decisions as (E1E2) has similarity value equal to 1, meanwhile (E3E4) also has the same similarity value. At this stage, AHC arbitrarily selects similarity value of (E3E4), so (E3E4) cluster is made (see Iteration 1 in Fig. 1).

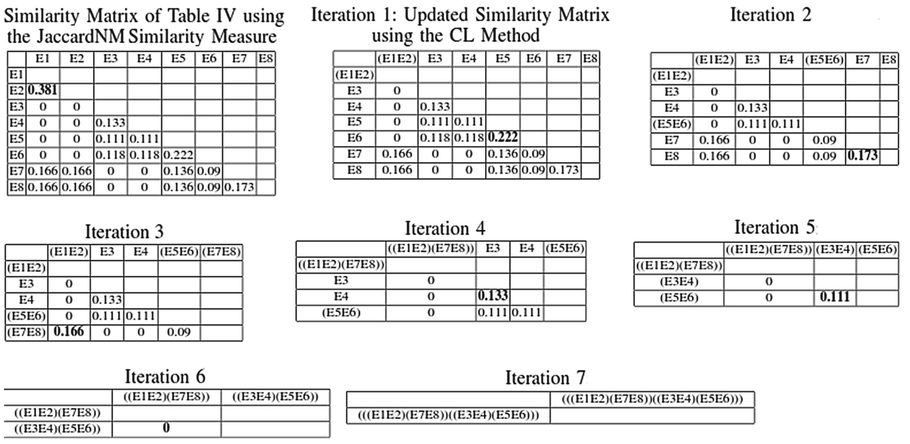
Table 4. Feature matrix

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13
E1	0	0	0	0	0	1	1	1	1	1	1	1	1
E2	0	0	0	0	0	1	1	1	1	1	1	1	1
E3	1	1	0	0	0	0	0	0	0	0	0	0	0
E4	1	1	0	0	0	0	0	0	0	0	0	0	0
E5	1	1	1	1	1	0	0	0	0	0	0	0	0
E6	1	1	1	1	0	0	0	0	0	0	0	0	0
E7	0	0	1	1	1	1	1	0	0	1	0	1	0
E8	0	0	1	1	1	1	0	1	1	0	1	0	0

The CL method is used to update the similarity values between the new cluster i.e. (E3E4) and all existing singleton clusters, and the updated similarity matrix (see Iteration 1 in Fig. 1). In the second iteration, AHC searches again for the maximum value in updated similarity matrix i.e. matrix in Iteration 1. This time it makes (E1E2) as a new cluster and updates its similarity values with all other existing clusters, as shown in Iteration 2 of Fig. 1. In iterations 3 and 4 it makes clusters of (E5E6) and (E7E8), respectively. In Iteration 3 it can be seen that there are two maximum values (i.e. 0.4), hence AHC may select either again. As stated before, AHC will select value that occurs later, therefore it makes cluster (E7E8). In the remaining iterations, AHC makes clusters of ((E3E4) (E5E6)), ((E1E2) (E7E8)) and (((E1E2) (E7E8)) ((E3E4) (E5E6))), as shown in Fig. 1

JaccardNM with CL Clustering Process. Now we apply the JaccardNM measure on the feature matrix given in Table 4, and get similarity matrix which can be seen in Fig. 2. The process for making clusters is the same as discussed in Subsect. 3.1. As per the AHC, the first cluster formed is (E1E2), second is (E5E6), third is (E7E8), fourth is ((E1E2) (E7E8)), fifth is (E3E4), sixth is ((E3E4) (E5E6)), and the last is (((E1E2) (E7E8)) ((E3E4) (E5E6))). The similarity matrices during iterations, i.e. from the first iteration to the seventh (n − 1) iteration, are given in Fig. 2. In each iteration, the CL method is used to update the similarity between newly formed and existing (singleton) clusters.

Discussion on the Results of Jaccard and JaccardNM Measures. In the previous two Subsects. 3.1 and 3.2, we observed that the Jaccard measure results in more number of clusters as compared to the JaccardNM measure. JaccardNM creates less number of arbitrary decisions as compared to the Jaccard. The JaccardNM produces results as expected because the main intuition of introducing this measure is to reduce the arbitrary decisions [6]. Hence, from these results we can easily conclude that the Jaccard has the strength to create more number of clusters, while the JaccardNM has the strength to reduce the number of arbitrary decisions.



Similarity Matrix of Table IV using the JaccardNM Similarity Measure

	E1	E2	E3	E4	E5	E6	E7	E8
E1								
E2	**0.381**							
E3	0	0						
E4	0	0	0.133					
E5	0	0	0.111	0.111				
E6	0	0	0.118	0.118	0.222			
E7	0.166	0.166	0	0	0.136	0.09		
E8	0.166	0.166	0	0	0.136	0.09	0.173	

Iteration 1: Updated Similarity Matrix using the CL Method

	(E1E2)	E3	E4	E5	E6	E7	E8
(E1E2)							
E3	0						
E4	0	0.133					
E5	0	0.111	0.111				
E6	0	0.118	0.118	**0.222**			
E7	0.166	0	0	0.136	0.09		
E8	0.166	0	0	0.136	0.09	0.173	

Iteration 2

	(E1E2)	E3	E4	(E5E6)	E7	E8
(E1E2)						
E3	0					
E4	0	0.133				
(E5E6)	0	0.111	0.111			
E7	0.166	0	0	0.09		
E8	0.166	0	0	0.09	**0.173**	

Iteration 3

	(E1E2)	E3	E4	(E5E6)	(E7E8)
(E1E2)					
E3	0				
E4	0	0.133			
(E5E6)	0	0.111	0.111		
(E7E8)	**0.166**	0	0	0.09	

Iteration 4

	((E1E2)(E7E8))	E3	E4	(E5E6)
((E1E2)(E7E8))				
E3	0			
E4	0	**0.133**		
(E5E6)	0	0.111	0.111	

Iteration 5

	((E1E2)(E7E8))	(E3E4)	(E5E6)
((E1E2)(E7E8))			
(E3E4)	0		
(E5E6)	0	**0.111**	

Iteration 6

	((E1E2)(E7E8))	((E3E4)(E5E6))
((E1E2)(E7E8))		
((E3E4)(E5E6))	**0**	

Iteration 7

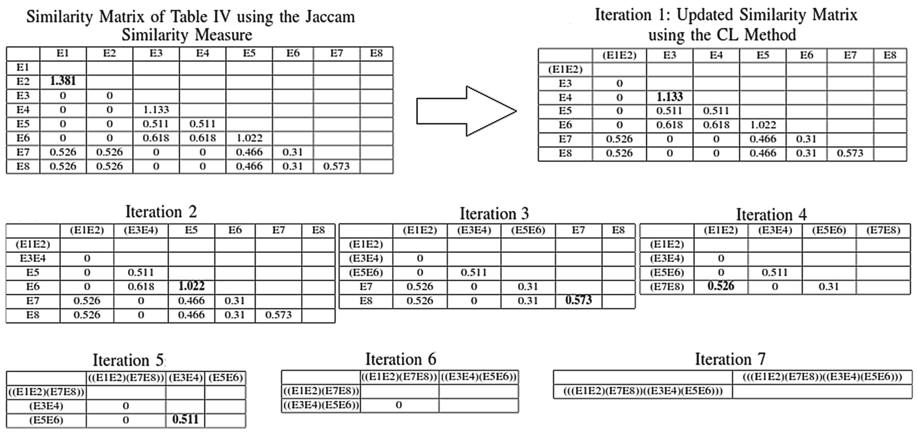
	(((E1E2)(E7E8))((E3E4)(E5E6)))
(((E1E2)(E7E8))((E3E4)(E5E6)))	

Fig. 2. The similarity matrix and iterations in clustering process using JaccardNM measure and CL method

3.2 The New Jaccam Measure

To combine the strengths of these existing similarity measures, the add operation is used to combine the existing similarity measures. The following subsections introduce our new measure and its analysis.

Addition of the Jaccard and JaccardNM Measures. The strengths of the Jaccard and JaccardNM measures can be combined by adding both the similarity measures to get the "Jaccam" measure. "Jaccam" is defined as:



Similarity Matrix of Table IV using the Jaccam Similarity Measure

	E1	E2	E3	E4	E5	E6	E7	E8
E1								
E2	**1.381**							
E3	0	0						
E4	0	0	1.133					
E5	0	0	0.511	0.511				
E6	0	0	0.618	0.618	1.022			
E7	0.526	0.526	0	0	0.466	0.31		
E8	0.526	0.526	0	0	0.466	0.31	0.573	

Iteration 1: Updated Similarity Matrix using the CL Method

	(E1E2)	E3	E4	E5	E6	E7	E8
(E1E2)							
E3	0						
E4	0	**1.133**					
E5	0	0.511	0.511				
E6	0	0.618	0.618	1.022			
E7	0.526	0	0	0.466	0.31		
E8	0.526	0	0	0.466	0.31	0.573	

Iteration 2

	(E1E2)	(E3E4)	E5	E6	E7	E8
(E1E2)						
E3E4	0					
E5	0	0.511				
E6	0	0.618	**1.022**			
E7	0.526	0	0.466	0.31		
E8	0.526	0	0.466	0.31	0.573	

Iteration 3

	(E1E2)	(E3E4)	(E5E6)	E7	E8
(E1E2)					
(E3E4)	0				
(E5E6)	0	0.511			
E7	0.526	0	0.31		
E8	0.526	0	0.31	**0.573**	

Iteration 4

	(E1E2)	(E3E4)	(E5E6)	(E7E8)
(E1E2)				
(E3E4)	0			
(E5E6)	0	0.511		
(E7E8)	**0.526**	0	0.31	

Iteration 5

	((E1E2)(E7E8))	(E3E4)	(E5E6)
((E1E2)(E7E8))			
(E3E4)	0		
(E5E6)	0	**0.511**	

Iteration 6

	((E1E2)(E7E8))	((E3E4)(E5E6))
((E1E2)(E7E8))		
((E3E4)(E5E6))	0	

Iteration 7

	(((E1E2)(E7E8))((E3E4)(E5E6)))
(((E1E2)(E7E8))((E3E4)(E5E6)))	

Fig. 3. The similarity matrix and iterations in clustering process using Jaccam measure and CL method

$$\begin{aligned} Jaccam &= Jaccard + JaccardNM \\ &= \frac{a(3(a+b+c)+d)}{(a+b+c)(2(a+b+c)+d)} \end{aligned} \tag{3}$$

The Example Feature Matrix and Jaccam Measure. To demonstrate the strengths of our new measure, we now apply the Jaccam similarity measure to the example feature matrix shown in Table 4. The corresponding similarity matrix using the Jaccam similarity measure is shown in Fig. 3. The CL linkage method is used to update the similarity matrix during the clustering process. We can see from similarity matrix that the Jaccam prioritizes the similarity values between pair of entities (E1E2) and (E3E4), as done by the JaccardNM in the similarity matrix given in the similarity matrix in Fig. 2. Hence, the decision to cluster the entities is no longer arbitrary. Entities E1 and E2 have a high value of similarity and are grouped first (see Iteration 1 in Fig. 3). Then in the subsequent iterations, the AHC makes clusters of (E3E4), (E5E6) and (E7E8). Note that in Iteration 3 the Jaccard measure creates arbitrary decisions while our new measure Jaccam does not, as shown in Figs. 1 and 3, respectively.

It is interesting to note that the Jaccam measure creates clusters as created by the Jaccard measure (i.e. 4 clusters) and similar to the JaccardNM measure, takes no arbitrary decisions. It can be inferred that our new measure has the strength to create larger number of clusters while reducing the arbitrary decisions taken by the AHC during the clustering process, so the Jaccam outperforms the existing similarity measure.

4 Experimental Setup, Results and Analysis

In this section, we present the test software systems used for experimental purposes and the setup of clustering process including the selection of assessment criteria. The assessment criteria are used to compare our new similarity measures with the well-established similarity measures for software clustering.

4.1 Datasets

To conduct the experiments we have used four software systems developed in Java, C and C++. These test software systems are different in their source code sizes and application areas. We use an open source software system, i.e. (1) Weka, an open source data mining software system used for data pre-processing, clustering, regression, classification, and visualization. In current study we use Weka version 3.4. All the proprietary software systems are developed using C++ programming language and they are: (1) FES, a fact extractor software system to extract the entities and features of software systems developed in Visual C++ programming language; (2) PLC, is a printer language converter into intermediate language; and (3) PLP, a parser software system used to parse a printer language. We obtained the extracted feature matrices of all these proprietary software systems from Muhammad et al. [2]. For all test systems classes are selected as entities.

4.2 Algorithms and Evaluation Criteria

For experiments we used CL and SL methods with Jaccam, Jaccard and JaccardNM measures. To evaluate the results, we consider authoritativeness (MoJoFM [14]), arbitrary decisions [13] and the number of clusters [3].

4.3 Arbitrary Decision

Table 5 presents the experimental results for all similarity measures. This table lists the average number of arbitrary decisions which are taken by the AHC using different similarity measures in each iteration. The first column in Table 5 shows the linkage methods. Similarity measures are shown in the second column while the arbitrary decision values for all test systems are given in the next four columns. The last column shows the average values for each similarity measure. The bold face values enclosed in parentheses indicate best values, while only bold face values represent the better values.

Table 5. Experimental results using arbitrary decisions

Algorithm	Measure	FES	PLC	PLP	Weka	Average
CL	Jaccam	10.28	**37.59**	**9.24**	695.42	94.07
	Jaccard	10.43	72.10	10.72	700.39	99.21
	JaccardNM	**10.26**	37.63	9.25	**695.13**	**94.03**
SL	Jaccam	3.00	(**35.90**)	(**1.85**)	374.69	51.93
	Jaccard	3.20	70.47	3.30	384.05	57.63
	JaccardNM	(**2.98**)	(**35.90**)	(**1.85**)	(**372.77**)	(**51.69**)

As can be seen from Table 5, our proposed similarity measure has reduced the arbitrary decisions for each linkage method. It is very interesting that Jaccam in most cases produce the similar number of arbitrary decisions as obtained by JaccardNM and also less than the Jaccard. The fact that both similarity measures, i.e. Jaccam and JaccardNM, have the ability to count all features, i.e. *a, b, c* and *d*. Therefore, both measures can clearly more distinguish the entities. Jaccard measure does not count *d* therefore can not distinguishes the entities and thus creates a large number of arbitrary decisions.

4.4 Number of Clusters

Table 6 shows the maximum number of non-singleton clusters, created by AHC during all iterations. The values enclosed in parentheses indicate best values, while only bold face values shows the better values. As can be seen from Table 6, that number of clusters created by Jaccam measure is higher than that created by JaccardNM measure and similar to that created by Jaccard measure.

Table 6. Experimental Results using Number of Clusters

Method	Measure	FES	PLC	PLP	Weka	Average
CL	Jaccam	(**10**)	10	(**12**)	(**55**)	(**10.88**)
	Jaccard	(**10**)	10	(**12**)	(**55**)	(**10.88**)
	JaccardNM	9	(**11**)	11	35	8.25
SL	Jaccam	**8**	**6**	**8**	**31**	**6.63**
	Jaccard	**8**	**6**	**8**	**31**	6.63
	JaccardNM	4	5	4	12	3.13

It can also be seen that for all software systems, our new measure substantially increased the number of clusters similar to the Jaccard measure. It is very interesting to note that the new measure integrating the Jaccard similarity measure as achieved equally large number of clusters as the Jaccard measure.

4.5 Authoritativeness

Authoritativeness finds the similarity between automated results (AR) and the authoritative decomposition (AD) prepared by a human expert. The AR should resemble the AD as much as possible for the better clustering results. In this study, the widely used MoJoFM [14], is utilized. MoJoFM finds the move and join operations to convert the AR into AD.

$$MoJoFM(AR, AD) = \left(1 - \frac{mno(AR, AD)}{max(mno(\forall AR, AD))}\right) * 100 \qquad (4)$$

where $mno(AR, AD)$ is the minimum number of 'move' and 'join' operations required to translate AR in to AD and $max(mno(\forall AR, AD))$ is the maximum of $mno(\forall AR, AD)$. MoJoFM results into a percentage of the similarity between two decompositions. A higher percentage indicates greater similarity between the AR and AD.

The MoJoFM values for the series of experiments are given in Table 7. This table shows the maximum MoJoFM values selected during the iterations of clustering process. The bold face values indicate the better values for a test system/method. The values enclosed in parentheses indicate best values in the Table 7. The average values for each similarity measure is shown in the last column of Table 7.

As can be seen from Table 7 that, in most of the cases our new measure outperform the existing ones. This is because in previous Subsects. 4.3 and 4.4, we shown that our new similarity measure results in smaller number of arbitrary decisions and larger number of clusters.

Table 7. MoJoFM Results for all Similarity Measures

Method	Measure	FES	PLC	PLP	Weka	Average
CL	Jaccam	**45.00**	61.54	(**65.67**)	**30.45**	(**40.53**)
	Jaccard	43.00	61.00	51.00	**30.45**	37.09
	JaccardNM	43.00	(**65.00**)	60.00	30.13	39.63
SL	Jaccam	**47.50**	**63.08**	**59.70**	22.12	**38.48**
	Jaccard	35.00	55.00	28.00	**23.08**	28.22
	JaccardNM	43.00	42.00	28.00	17.31	26.06

5 Conclusion

This paper presents a new binary similarity measures (namely Jaccam) for software clustering. This measure integrates the strengths of the following existing binary similarity measures: Jaccard and JaccardNM. An example case study is used to show how our new measure integrates strengths of the existing similarity measures, i.e., reducing the arbitrary decisions and increasing the number of clusters. The Jaccam and existing binary similarity measures are assessed using four different software systems implemented in different programming languages.

One of the most remarkable strengths from the integration of the existing binary similarity measures is that Jaccam results in the large number of clusters which results in improving the authoritativeness. The new measure also reduces arbitrary decisions to lessen the complications of making clusters during the clustering process.

Acknowledgment. The authors would like to thank Office of Research, Innovation, Commercialization and Consultancy Office (ORICC), Universiti Tun Hussein Onn Malaysia (UTHM) for financially supporting this research under the Postgraduates Incentive Grant (GIPS) vote no. U063.

References

1. Shtern, M., Tzerpos, V.: Methods for selecting, improving software clustering algorithms. Softw. Pract. Exp. **44**(1), 33–46 (2014)
2. Muhammad, S., Maqbool, O., Abbasi, A.Q.: Evaluating relationship categories for clustering object-oriented software systems. IET Softw. **6**(3), 260 (2012)
3. Maqbool, O., Babri, H.: Hierarchical clustering for software architecture recovery. IEEE Trans. Softw. Eng. **33**(11), 759–780 (2007)
4. Shtern, M., Tzerpos, V.: On the comparability of software clustering algorithms. In: 2010 IEEE 18th International Conference on Program Comprehension, pp. 64–67. IEEE, June 2010
5. Cui, J.F., Chae, H.S.: Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems. Inf. Softw. Technol. **53**(6), 601–614 (2011)

6. Naseem, R., Maqbool, O., Muhammad, S.: An improved similarity measure for binary features in software clustering. In: 2010 Second International Conference on Computational Intelligence, Modelling and Simulation, pp. 111–116. IEEE, September 2010
7. Shtern, M., Tzerpos, V.: Clustering methodologies for software engineering. Adv. Softw. Eng. **2012**, 1–18 (2012)
8. Maqbool, O., Babri, H.: The weighted combined algorithm: a linkage algorithm for software clustering. In: Eighth European Conference on Software Maintenance and Reengineering, pp. 15–24. IEEE (2004)
9. Saeed, M., Maqbool, O., Babri, H., Hassan, S., Sarwar, S.: Software clustering techniques and the use of combined algorithm. In: Seventh European Conference on Software Maintenance and Reengineering, pp. 301–306. IEEE Computer Society (2003)
10. Naseem, R., Maqbool, O., Muhammad, S.: Improved similarity measures for software clustering. In: 2011 15th European Conference on Software Maintenance and Reengineering, pp. 45–54. IEEE, March 2011
11. Andritsos, P., Tzerpos, V.: Information-theoretic software clustering. IEEE Trans. Softw. Eng. **31**(2), 150–165 (2005)
12. Bauer, M., Trifu, M.: Architecture-aware adaptive clustering of OO systems. In: Eighth European Conference on Software Maintenance and Reengineering, 2004, CSMR 2004, Proceedings, pp. 3–14 (2004)
13. Naseem, R., Maqbool, O., Muhammad, S.: Cooperative clustering for software modularization. J. Syst. Softw. **86**(8), 2045–2062 (2013)
14. Wen, Z., Tzerpos, V.: An effectiveness measure for software clustering algorithms. In: Proceedings, 12th IEEE International Workshop on Program Comprehension, 2004, pp. 194–203. IEEE (2004)