Fuzzy Inf. Eng. (2014) 6: 263–278

Available online at www.sciencedirect.com
ScienceDirect
Fuzzy Information and Engineering
http://www.elsevier.com/locate/fiae

ORIGINAL ARTICLE

Improving n-Similarity Problem by Genetic Algorithm and Its Application in Text Document Resemblance

M. Mirhosseini · M. Mashinchi · H. Nezamabadi-pour

Received: 5 May 2013/ Revised: 22 June 2014/
Accepted: 22 October 2014/

Abstract In this paper, some methods of similarity measures between objects are presented with their properties reviewed. The study is conducted to propose a new method based on genetic algorithm in order to reduce the time complexity of finding n most similar objects among the huge number of objects. This method is tested on two applications. The former aims at finding the most similar residents in a condominium, and the latter deals with finding the most similar n-groups of text documents out of a great dataset. The simulation results show that the proposed method can efficiently improve the order of time complexity especially for the second application.

Keywords n-Similarity · Genetic algorithm · Similarity measure

1. Introduction

In literature, there are different definitions for similarity concept in various fields. For example definition of similarity in geometry differs from that of chemistry or psychol-

M. Mirhosseini (✉)
Department of Computer Science, Faculty of Mathematics and Computing, Higher Education Complex of Bam, Bam, Kerman, Iran
email: mirhosseini@bam.ac.ir
M. Mashinchi
Department of Statistics, Faculty of Mathematics and Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran
H. Nezamabadi-pour
Department of Electrical Engineering, Faculty of Engineering, Shahid Bahonar University of Kerman, Kerman, Iran
Peer review under responsibility of Fuzzy Information and Engineering Branch of the Operations Research Society of China.

http://dx.doi.org/10.1016/j.fiae.2014.12.001

ogy. In general, similarity has been defined as the degree of resemblance between two or more concepts or objects [1, 2]. Each object, usually can be represented by special properties composing a property vector of that object. The similarity degree between two objects can be obtained by comparing their property vectors. Similarity value is usually expressed as a real number within the intervals $[0, 1]$ or $[-1, 1]$. The values 0 or -1 specify that objects are completely different, whereas 1 value expresses that objects are identical [2]. Different types of similarity measures have been proposed by researchers, including feature contrast model [3], information content [4], mutual information [5], Dice coefficient [6], cosine coefficient [6] and distance-based measurements [7]. Some classic similarity and dissimilarity measures for clustering have been discussed in [8]. Sixty seven similarity measures in information retrieval have been compared in [9].

The comparison of similarity measures in fuzzy scope has been discussed in [10]. This type of measure, takes into account the intersection of two fuzzy sets, where their values are measured based on the geometric distance model, set-theoretic approach and matching function approaches.

Another similarity measure between two fuzzy sets can be found in [11, 12]. The idea in these jobs is that if a weak intersection exists between two fuzzy sets, then their distance will be greater. In [11] the relative sigma count between fuzzy sets is used to define two other similarity measures between fuzzy sets. The form of similarity discussed in [12] is very close to correlation-based approaches which precisely can consider it as a dissimilarity measure. Cosine similarity measure and a weighted cosine similarity measure between intuitionistic fuzzy sets (IFS) are proposed based on the concept of the cosine similarity measure for fuzzy sets [13].

All these similarity measures have been defined between just two objects or sets. In many cases and applications the pair-wise similarities between objects are known, and it is needed to find group of $n \geq 3$ objects with the most similarity, which may be useful in object clustering or classification methods.

For the first time, the concept of similarity is generalized among n objects by Keshavarzi et al. [1, 2]. It could work with various known 2-similarity measures. But despite theoretically proved background of the approach, it could not be applied to great search spaces.

In this paper, by using genetic algorithm, we improve the n-similarity problem, to compute the similarity among n objects for $n \geq 3$, or to find a group of n objects which has the most similarity within a dataset. Also we will apply this method to digital text documents to search the groups with the most similarity values.

With significant growing of digital documents, information management and search become a crucial problem [14]. Searching and finding the similar documents has special importance in text document management [15]. Different similarity measures have been suggested in the text retrieval scope which is used for computing the similarity degree between the two documents and have been applied to text clustering or text classification. In [16] several similarity measures have been introduced and compared in data mining. In [17, 18] the effect of different similarity measures such as cosine similarity measure, Jaccard coefficient, Euclidean similarity and Pearson correlation coefficient on text clustering is discussed. In addition, in [17] an asym-

metry similarity measure named averaged Kullback-Leibler divergence is compared with mentioned measures on several datasets. In [19], a semantic similarity measure based on topic maps representation is proposed and compared with previous methods. Topic maps which are trees representations of documents and the similarity between a pair of documents, is computed as a correlation between the sub-trees.

The longest common subsequence method (LCS) has been proposed in [20] that obtains the maximum-length common subsequence of two text strings. In this method, a sequence $Z = z_1, \cdots, z_m$ is a subsequence of $X = x_1, \cdots, x_k$ if a strictly increasing sequence $i_1, \cdots, i_m$ in indices of X exist, such that for all $j = 1, \cdots, m$, $x_{i_j} = z_j$. By considering this definition, two strings are more similar if the maximum-length of their common subsequence be larger. In comparison with LCS, a formal recursive definition of n-gram similarity in strings by an efficient algorithm has been described in [21].

In [15] a novel similarity measure for fuzzy text clustering has been presented. Authors in [22] have focused on finding out the most similar documents using shingling technique in big data; also the Jaccard coefficient to judge the degree of similarity between the text documents has been discussed.

As mentioned above, the problem with all these measures is used to compute the value of similarity just between two objects. In this paper, we improve the proposed n-similarity theory of Keshavarzi et al. [1, 2].

To solve the n-similarity problem by using Keshavarzi′s method [1, 2], computing $2, 3, \cdots, (n-1)$-similarity values are needed, so this method is practically so time-consuming, and there is the need to improving the performance of the algorithm. We optimize this method by exploiting the genetic algorithm, so that it is independent of computing $2, 3, \cdots, (n-1)$-similarity values and just uses 2-similarity values. It will be proved that to find out the most n similar objects among m objects, the complexity of the algorithm decline from $O(m^n)$ in Keshavarzi′s method [1, 2], to $O(n^2)$ in our proposed method by using genetic algorithm. Therefore, the proposed method is able to handle huge datasets in reasonable time.

We apply the improved n-similarity measure for resemblance text documents. We examine this method on the text document dataset, in order to find the most similar n documents among a large corpus. Our analysis and simulation results show that our proposed approach by genetic algorithm can dramatically decline the complexity of n-similarity problem, especially in great datasets in comparison with previous method.

The remainder of the paper is organized as follows. Section 2 gives some preliminaries that are used in this paper. Section 3 describes our approach to improve the n-similarity problem by genetic algorithm. Some genetic algorithm concepts that are needed have been explained in this section. Finally, in Section 4 simulation results have been shown and discussed and a formal comparison between two methods has been presented.

2. Preliminaries

In this section, we use some notations that are needed through this study.

Definition 2.1 [1, 2] *Triangular norm (T-norm) is a function* $T : [0,1] \times [0,1] \to [0,1]$ *which for all* $x, y, w, z \epsilon [0,1]$ *satisfies the following conditions:*

i. *Commutativity:* $T(x, y) = T(y, x)$;

ii. *Monotonicity:* $T(x, y) \leq T(w, z) if\, x \leq w, and\, y \leq z$;

iii. *Associativity:* $T(x, T(y, w)) = T(T(x, y), w)$;

iv. *Boundary:* $T(x, 0) = 0, T(x, 1) = x$.

The minimum T-norm is one of the most important instances of T-norms which has been used in [1, 2, 23] for generalization of similarity measures $T_{\min}(x, y) = \min(x, y)$.

Definition 2.2 [1] *A 2-similarity on a domain U is a function* $S : U \times U \to [0, 1]$ *such that the following conditions are satisfied:*

i. *Reflexivity: for any* $x \epsilon U, S(x, y) = 1$;

ii. *Symmetry: for any* $x, y \epsilon U, S(x, y) = S(y, x)$;

iii. *Transitivity: for any* $x, y, z \epsilon U, S(x, z) \geq S(x, y) \wedge S(y, z)$, *where* $\wedge$ *is a minimum operator.*

If the implication $S(x, y) = 1 \Rightarrow x = y$ is confirmed, then it can be said that S is strict.

Definition 2.3 [1, 2] *A 3-similarity on a domain U is a function* $S : U \times U \times U \to [0, 1]$ *such that the following conditions be satisfied:*

i. *Reflexivity: for any* $x \epsilon U, S(x, x, x) = 1$;

ii. *Symmetry: for any* $x_1, x_2, x_3 \epsilon U, S(x_1, x_2, x_3) = S(x_{i1}, x_{i2}, x_{i3})$ *where* $(i1, i2, i3)$ *is an arbitrary permutation of* $(1, 2, 3)$;

iii. *Transitivity property: for any* $t, x_1, x_2, x_3 \epsilon U, S(x_1, x_2, x_3) \geq S(t, x_2, x_3) \wedge S(x_1, t, x_3) \wedge S(x_1, x_2, t)$, *where* $\wedge$ *is minimum T-norm.*

If the implication $S(x_1, x_2, x_3) = 1 \Rightarrow x_1 = x_2 = x_3$ is confirmed, then it is said that S is strict. In [23] Minimum T-norm has been used for generalizing the 2-similarity to 3-similarity as defined in (1). It has been proved that this equation satisfies in Definition 2.3.

$$S_3(x, y, z) = \min\{S_2(x, y), S_2(y, z), S_2(x, z)\}. \tag{1}$$

Using the definitions of 2-similarity and 3-similarity, the notion of n-similarity is defined as follow:

Definition 2.4 [1, 2] $S : U \times U \times \cdots \times U \to [0, 1]$ *is an n-similarity if the following properties are satisfied:*

i. *Reflexivity: for any* $x \epsilon U, S(x, x, \cdots, x) = 1$;

ii. *Symmetry:* $S(x_1, x_2, \cdots, x_n) = S(x_{i1}, x_{i2}, \cdots, x_{in})$ *for all permutations* $(i1, i2, \cdots, in)$ *of* $(1, 2, \cdots, n)$;

iii. *Transitivity: for all* $x_1, x_2, \cdots, x_n, z \epsilon U, S(x_1, x_2, \cdots, x_n) \geq \min\{S(z, x_2, \cdots, x_n), \cdots, S(x_1, x_2, \cdots, x_{n-1}, z)\}$.

In [1, 2] it has been proved that the n-similarity can be obtained from an $(n-1)$-similarity, while satisfies the mentioned properties.

If S_{n-1} is an $(n-1)$-similarity on U and $x_1, x_2, \cdots, x_n \epsilon U$, then

$$\begin{aligned} &S_n(x_1, \cdots, x_n) \\ &= \min\{S_{n-1}(x_2, x_3, \cdots, x_n), S_{n-1}(x_1, x_3, \cdots, x_n), \cdots, S_{n-1}(x_1, x_2, \cdots, x_{n-1})\}. \end{aligned} \tag{2}$$

Definition 2.5 [1, 2] *Let U be a set and $S : U \times U \times \cdots \times U \to [0, 1]$ be an n-similarity on U. Then for any $\lambda \epsilon [0, 1]$, the n-relation $\cong_{S,\lambda}$ in U is defined as $(x_1, x_2, \cdots, x_n) \epsilon \cong_{S,\lambda}$ if $S(x_1, x_2, \cdots, x_n) \geq \lambda$. Then set $\cong_{S,\lambda}$ is called cut of level λ of S or λ-cut of S.*

The pseudo codes of 3-similarity, 4-similarity and n-similarity have been shown in Algorithms 1, 2 and 3, respectively, based on the paper by Keshavarzi et al. [1, 2].

Algorithm 1 Three-similarity algorithm

Input: 2-sim matrix, λ
Output: 3-sim matrix and the most 3-similar cluster, λ-cut set
For $i = 1$ to size of dataset
For $j = i + 1$ to size of dataset
For $k = j + 1$ to size of dataset
3-$sim(i, j, k)$=min $\{2\text{-}sim(i, j), 2\text{-}sim(i, k), 2\text{-}sim(j, k)\}$;
If 3-$sim(i, j, k) \geq \lambda$
Insert 3-$sim(i, j, k)$ in λ-cut set
End for of i, j and k
Write maximum element of 3-sim as the most 3-similar cluster
End

Algorithm 2 Four-similarity algorithm

Input: 3-sim matrix, λ
Output: 4-sim matrix and the most 4-similar cluster, λ-cut set
For $i = 1$ to size of dataset
For $j = i + 1$ to size of dataset
For $k = j + 1$ to size of dataset
For $l = k + 1$ to size of dataset
4-$sim(i, j, k, l)$=min$\{3\text{-}sim(i, j, k), 3\text{-}sim(i, j, l), 3\text{-}sim(i, k, l), 3\text{-}sim(j, k, l)\}$;
If 4-$sim(i, j, k, l) \geq \lambda$
Insert 4-$sim(i, j, k, l)$ in λ-cut set
End for of i, j, k and l
Write maximum element of 4-sim as the most 4-similar cluster
End

Algorithm 3 n-similarity algorithm

Input: $(n-1)$-sim matrix, λ
Output: n-sim matrix and the most n-similar cluster, λ-cut set
For $i_1 = 1$ to size of dataset
For $i_2 = i_1 + 1$ to size of dataset
$\vdots$
For $i_n = i_{n-1}$ to size of dataset
n-$sim(i_1, i_2, \cdots, i_n)$=min $\{(n-1)$-$sim(i_2, i_3, \cdots, i_n)$,$(n-1)$-$sim(i_1, i_3, \cdots, i_n), \cdots, (n-1)$-$sim(i_1, i_2, \cdots, i_{n-1})\}$;
If n-$sim((i_1, i_2, \cdots, i_n)) \geq \lambda$
Insert n-$sim(i_1, i_2, \cdots, i_n)$ in λ-cut set
End for of i_1 to i_n
Write maximum element of n-sim as the most n-similar cluster
End

In fact, by this way, to find the most n-similar group, all possible $2, 3, \cdots n$-similarity values among objects should be computed, then the n-group with maximum similarity can be obtained.

Obviously, such deterministic method is highly time-consuming and its search space rapidly grows when the n or the dataset size increases. This problem leads to significant decrease in algorithm performance.

In order to overcome this problem, we utilize the genetic algorithm which significantly improves the complexity of the problem. It is proved in Lemma 1 that our proposed method by genetic algorithm just uses 2-similarity values as input. Furthermore, the size of the dataset and n does not directly affect on its algorithm. This is for the sake of genetic algorithms nature that just search in some candidate solutions instead of searching all possible solutions [24].

3. Genetic Algorithm

Genetic algorithms (GAs) as a subclass of evolutionary algorithms are random search methods and are often used to optimize problems. Its principle is inspired by natural selection and biological genetic to survive the fittest [25, 26].

Biological genetic is simulated by an iterative process called genetic algorithm described afterward. First, an initial population is generated randomly. Each individual in the population is represented as a bit string. Each individual is called chromosome and each chromosome represents a possible solution to the problem. Then, fitness evaluation is computed for each decoded solution. Crossover and mutation are executed on the current generation and produce a new generation. This process is repeated for a certain number of times or until stopping criterion be satisfied [27]. In the following, genetic operators are explained in details.

3.1. Representation

The first step in GA is setting a link between the problem context and problem-solving space. The binary method used in GAs is one of the earliest representations. In

this approach, each solution is encoded as a bit-string. For a particular application, the length of bit-string should be first determined, so all possible solutions can be represented by bit-string, and all possible bit-strings produce valid solutions [27].

3.2. Evaluation

Evaluation function is a procedure that assigns a quality measure to each individual. The evaluation function is usually called fitness function in GAs. This function depends on the type of optimization problem [27].

3.3. Selection

Selection is a process that determines which solutions influence on the next generation to find optimum solutions in a short time. Its goal is to select the best solutions in the population and to drive them into mating pool with this hope that by combination of them, higher fitness off-spring will be produced.

The roulette wheel is a popular selection method that is often used in GAs. In this method, the individuals with a higher fitness value, have higher chance to select. The selection probability of each individual is its fitness divided by sum of the fitness of all individuals in the current population. By using these probabilities, cumulative distribution function (CDF) is calculated for each individual in the population. A random number r between 0 and 1 is chosen. The individual with the biggest CDF value which is equal or greater than r, is selected and inserted in the mating pool as a parent of the next generation. This process should be repeated N times, where N is the size of the population [24].

3.4. Crossover

Crossover is the main distinguishing feature of GAs. Single-point is a simple type of crossover that is often used. In this method, two individuals are randomly selected from the mating pool. A random number (less than or equal to the chromosome length) is generated as the crossover position. The parents′ two parts after the crossover position are exchanged to generate two new off-spring and then are passed along to the next generation. The crossover operation is done by a pre-defined probability that describes how often crossover should be performed [24]. Figure 1 illustrates the single point crossover.

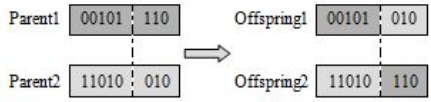

Fig. 1 Single point crossover

3.5. Mutation

The mutation operator is applied to each string after the crossover operation. This unary genetic operator produces a child by changing randomly selected gene in binary representation. Although this operator does not guarantee discovering a global

optimum solution, it often provides a sub-optimum but a nearly global optimal solution. In fact, mutation randomly walks through the vicinity of the candidate solutions and manipulates some solutions to stop the algorithm from being trapped in the local optimum. A simple mutation often used in binary representation is done by flipping the value of some randomly selected bits with special mutation rate [24, 25, 27].

Crossover and mutation operators produce a set of new candidate solutions based on their fitness. These solutions are moved to the next generation. All these mentioned operations are performed together in an iterative manner until a termination condition is satisfied [27].

4. The Proposed Method

Consider all m objects in the dataset have been numbered from 1 to m. The purpose is to find the most n-similar cluster among these m objects. We represent each solution as a string of size $n \times \lceil \log(m) \rceil$ bits, because each cluster should be consisted of n variables, where each variable ranges between 1 to m. Such strings are the candidate solutions named chromosomes [25].

Figure 2 illustrates an instance of the chromosomes where $Li = \lceil \log(m) \rceil$, and $\lceil \rceil$ stands for ceiling bracket operation. An initial population is randomly generated in a $N \times (n \times \lceil \log(m) \rceil)$ matrix of 0 and 1s, where N is the size of the population.

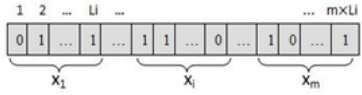

Fig. 2 Chromosome representation as bit string

Each variable can be decoded to real numbers by (3). It will be normalized by (4), and then the relation (5) converts it to a number between $[1, m]$. Finally, the result should be rounded to integer numbers in our problem.

$$(x_i)_{10} = \sum_{k=0}^{Li-1} bit(k) \times 2^k, \tag{3}$$

$$x_i^n = \frac{(x_i)_{10}}{2^{Li} - 1}, \tag{4}$$

$$x_i^{real} = 1 + x_i^n \times (m - 1). \tag{5}$$

When a new population initialized or created, the fitness values should be computed for each candidate solution [25]. The fitness value is calculated for each candidate solution by finding the minimum similarity among similarity values of all pairwise variables of each candidate solution. Hereby a 2-similarity matrix will be obtained

by a special similarity measure among all pair objects. That method is equivalent to Keshavarzi's n-similarity definition as it is shown in the following lemma.

Lemma 1 *Let* $S_n : U\times U\times\cdots\times U \to [0,1]$ *be n-similarity on U, where* $U\times U\times\cdots\times U$ *is the n copies of Cartesian product of U. Also* $S_{n-1},\cdots,S_4,S_3$ *and* S_2 *are* $(n-1)$*-similarity,* $\cdots$*,* 4*-similarity,* 3*-similarity and* 2*-similarity, respectively.*

proof According to (2), n-similarity is computed as follow:

$$S_n(x_1,\cdots,x_n) = \min\{S_{n-1}(x_2,x_3,\cdots,x_n), S_{n-1}(x_1,x_3,\cdots,x_n),\cdots,S_{n-1}(x_1,x_2,\cdots,x_{n-1})\}.$$

In the same way,

$$S_{n-1}(x_1,\cdots,x_{n-1}) = \min\{S_{n-2}(x_3,\cdots,x_n), S_{n-2}(x_1,x_4,\cdots,x_n),\cdots,S_{n-2}(x_1,x_2,\cdots,x_{n-2})\},$$

$$\vdots \qquad\qquad \vdots$$

$$S_4(x_1,x_2,x_3,x_4) = \min\{S_3(x_2,x_3,x_4), S_3(x_1,x_3,x_4), S_3(x_1,x_2,x_4), S_3(x_1,x_2,x_3)\},$$

and

$$S_3(x_1,x_2,x_3) = \min\{S_2(x_2,x_3), S_2(x_1,x_3), S_2(x_1,x_2)\}.$$

Recursively, by replacing S_3 in S_4, S_4 in S_5,$\cdots$ and S_{n-1} in S_n and removing repetitive elements, S_n will be obtained just in terms of S_2 and can be computed regardless of the previous $(n-1)$ similarity matrices. So n-similarity of n objects can be obtained by taking the minimum of 2-similarities of all possible pairwise of n objects.

Algorithm 4 represents a fitness function process that computes the similarity value of each candidate solution. It finds the maximum fitness and λ-cut set during the generations or iterations of the algorithm.

Algorithm 4 Fitness function of GA

Input: 2-sim matrix, λ
Output: 3-sim matrix and the most 3-similar cluster, λ-cut set
For $i = 1$ to N (size of population)
  For $j = 1$ to n
    For $k = j+1$ to n
      $Func(i)$= 2-$sim(decoded-Value(i,j), decoded-Value(i,k))$;
    End of for k
  End of for j
$fitness\ (i)$ = minimum $(Func(i))$;
If $fitness(i) \geq \lambda$
  Insert $decoded-Values(i)$ in λ-cut set.
End of for i
Find the maximum fitness of this population.
End

After evaluating each solution, the next step in this process is selection. This step selects better solutions in the population based on their fitness value, and places them in mating pool for recombination purpose. In this study, we have used a well known selection named as roulette wheel which has been explained in Section 3.3. The solu-

tions selected and entered in the mating pool are combined in the crossover process. We apply single-point crossover in this work. This method has been described in Section 3.4. Finally, the mutation operator is applied to avoid the algorithm falling into local optimum. As mentioned in Section 3.5, mutation is done by flipping the value of some bits randomly with mutation probability p_m.

After these operations, the new generated population is replaced by the old one. This algorithm is performed repeatedly until the termination condition is satisfied. This algorithm returns the most optimum solution. λ-cut set means the solutions that their fitness value is equal or more than a pre-defined level λ. So increasing the number of algorithm iterations can guaranty obtaining more perfect λ-cut set.

5. Experimental Results

In many cases, it is necessary to find n groups of m objects that their similarity is equal or more than a special value or we need to find the most n-similar group out of a dataset.

In this section, we test our approach on two datasets. The first one is the dataset of 2-similarity between 20 families used in [1]. The problem is arrangement of residents in a condominium such that the manager aims to choose some (two, three or more) of most similar residents for some special purposes. We apply this dataset in order to emphasis on correctness of our proposed approach. Further, we will apply more realistic and huge dataset in text similarity application. In this case, previous methods of [1] are unable to handle such datasets.

Application 1 (Finding similar groups of families)

The aim is to find the most 3-similar groups and λ-cut set of 20 families that their pairwise similarities have been shown in Figure 3. These residents similarities have been obtained based on some criteria [1]. We applied both Algorithm 1 (Keshavarzi′s method) and our proposed approach on this dataset. Parameters of our GA method were: $N = 100, P_c = 0.9, P_m = 0.05, ITER = 100$ and $\lambda = 0.6$; where N represents population size, P_c is the crossover probability, P_m is the mutation probability, $ITER$ is the number of generations or iteration of the algorithm and λ is the similarity level.

Both methods are used to obtain maximum similarity 0.6 on the families F_2, F_{15}, F_{16} and of course λ-cut set just contains this solution for $\lambda = 0.6$. The experiments performed on GA, had been repeated for 25 times. Figure 3 shows the average fitness function of all experiments during 100 generations. It is essential that the algorithm ignore producing those solutions that have frequent objects such as (x, x, y), (x, x, x) and etc. For this, since the 2-similarity matrix is symmetric and the values on its main diameter are one, we assign the values below and on the main diameter of 2-similarity matrix to zero.

Although for small size datasets, Keshavarzi′s method [1, 2] can find solutions faster than GA, but for huge datasets it fails to respond in acceptable time or they may not be able to find any solutions at all.

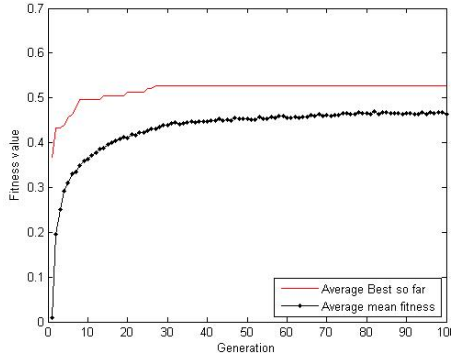

Fig. 3 Average best so far and average mean fitness of 3-similarity of families during generations in 25 running of the algorithm

Table 1: The similarity value of families [1].

Family	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
F_1	1.0	0.4	0.4	0.1	0.1	0.1	0.0	0.2	0.2	0.4	0.4	0.4	0.0	0.4	0.4	0.4	0.0	0.0	0.0	0.0
F_2	0.4	1.0	0.4	0.1	0.1	0.1	0.0	0.2	0.2	0.4	0.4	0.4	0.0	0.4	0.6	0.8	0.0	0.0	0.0	0.0
F_3	0.4	0.4	1.0	0.1	0.1	0.1	0.0	0.2	0.2	0.4	0.4	0.4	0.0	0.4	0.4	0.4	0.0	0.0	0.0	0.0
F_4	0.1	0.1	0.1	1.0	0.2	0.8	0.0	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0
F_5	0.1	0.1	0.1	0.2	1.0	0.2	0.0	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0
F_6	0.1	0.1	0.1	0.8	0.2	1.0	0.0	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0
F_7	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
F_8	0.2	0.2	0.2	0.1	0.1	0.1	0.0	1.0	0.4	0.2	0.2	0.2	0.0	0.2	0.2	0.2	0.0	0.0	0.0	0.0
F_9	0.2	0.2	0.2	0.1	0.1	0.1	0.0	0.4	1.0	0.2	0.2	0.2	0.0	0.2	0.2	0.2	0.0	0.0	0.0	0.0
F_{10}	0.4	0.4	0.4	0.1	0.1	0.1	0.0	0.2	0.2	1.0	0.8	0.4	0.0	0.4	0.4	0.4	0.0	0.0	0.0	0.0
F_{11}	0.4	0.4	0.4	0.1	0.1	0.1	0.0	0.2	0.2	0.8	1.0	0.4	0.0	0.4	0.4	0.4	0.0	0.0	0.0	0.0
F_{12}	0.4	0.4	0.4	0.1	0.1	0.1	0.0	0.2	0.1	0.4	0.4	1.0	0.0	0.8	0.4	0.4	0.0	0.0	0.0	0.0
F_{13}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.4	0.4	0.4	0.4
F_{14}	0.4	0.4	0.4	0.1	0.1	0.1	0.0	0.2	0.2	0.4	0.4	0.8	0.0	1.0	0.4	0.4	0.0	0.0	0.0	0.0
F_{15}	0.4	0.6	0.4	0.1	0.1	0.1	0.0	0.2	0.2	0.4	0.4	0.4	0.0	0.4	1.0	0.6	0.0	0.0	0.0	0.0
F_{16}	0.4	0.8	0.4	0.1	0.1	0.1	0.0	0.2	0.2	0.4	0.4	0.4	0.0	0.4	0.6	1.0	0.0	0.0	0.0	0.0
F_{17}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	1.0	0.4	0.6	0.4
F_{18}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.4	1.0	0.4	0.8
F_{19}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.6	0.4	1.0	0.4
F_{20}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.4	0.8	0.4	1.0

Application 2 (n-similarity between text documents)

In many studies, the various kinds of 2-similarity approaches have been compared and used in the text document clustering. Here, we are going to use n-similarity concept to find the most similar n group of text documents in a corpus.

We applied Re0 [28] dataset that contains 1504 documents of newspaper articles [17]. In text document, there are a lot of words that do not describe the topic of text such as *a*, *the*, *is*, *are*, etc. These kinds of words are called stop-words that should be removed from the text. We have used a pre-supplied list used in the Weka machine learning workbench that contains 527 stop-words. Also in text documents, there are words that are thematically similar but have a different morphological concept. These words should be mapped into their stem to treat as a single word. For example, the words computing, computation, have a similar concept. Although their morphology is different, but these words are stemmed to compute [17]. We used Porters suffix-stripping algorithm [29].

We modeled the documents as a bag of words and used word-by-word comparison. If $T = \{t_1, t_2, \cdots, t_n\}$ is the set of all occurred terms in the corpus, each document can be represented as a feature vector $d_i = \{w_{1i}.w_{2i}, \cdots, w_{ni}\}$, where w_{ki} is the weight of term t_k in document d_i. We used several weighting schemas. The first was Boolean weighting which 1 is representative of occurrence and 0 non-occurrence of term in document. Second schema was the term frequency of each term in each document. The other weighting schema was TF-IDF weighting which is defined as:

$$tfidf(d,t) = tf(d,t) \times \log(\frac{|D|}{df(t)}), \tag{6}$$

where $tf(d,t)$ is the frequency of term t in document d, D is the number of documents in the corpus and $df(t)$ is the number of documents that term t occurs in. After weighting step, the similarity between all pairs of documents should be computed in a composed similarity matrix [16].

There are various similarity measures for TF-IDF representation that are explained and compared in [17] on several datasets. The experiment on Re0 [28] shows that the cosine similarity measure is more suitable for this dataset. It computes the similarity between documents d_i, d_j as follows [16]:

$$cosine - sim(d_i, d_j) = \frac{\sum_{k=1}^{n} w_{ki} w_{kj}}{\sqrt{\sum_{k=1}^{n} w_{ki}^2 \sum_{k=1}^{n} w_{kj}^2}}. \tag{7}$$

After obtaining the similarity matrix, this matrix is entered as an input to explained genetic algorithm in Section 3. The output of this algorithm is the most n-similar group(s) of objects, and also the set of solutions with n-similarity values greater than predefined λ.

Experiments from the suggested method have been done on a system with 4GB memory and CPU speed of 2.27 GH. Text document preprocessing step has been im-

plemented by Java programming language and genetic algorithm step has been simulated in Matlab R2010a. The results have been reported through the 25 independent running of the algorithm. The parameters of the algorithm have been set as: population $size = 500$, $ITER = 500$, crossover population=0.9, mutation probability=0.005 and λ=0.7.

Table 2 illustrates a summary of n-similarity experiment results on Re0 dataset [28], as n varies from 3 to 6. In the second and the third columns of this table, average of the best and mean of the fitness values has been computed during 500 generations respectively. In the fourth, fifth and sixth columns the average, median and maximum of the best fitness through 25 running of the algorithm has been reported. The seventh column shows the document names of Re0 dataset with the most n-similarity value. The next column has counted the number of solutions with n-similarity values more than λ=0.7 and finally the two latest columns show the execution time of our proposed algorithm and Keshavarzi′s method respectively. These two columns show the superior preference of our suggested method by using genetic algorithm in comparison with keshavarzi's method in [1, 2] which is unable to find the solution in large dataset.

Fig. 4 to Fig. 7 show the average of best and mean fitness value of 3 to 6-similarity during the generations respectively.

Table 2: The n-similarity results on Re0 dataset.

n	Average best so far	Average mean fitness	Average of maximum similarities	Median of maximum similarities	Maximum similarity	The best solution	Average of the size of λ-cut set	The execution time of our method (s)	The execution time of [2] (s)
3	0.9841	0.7758	0.9861	0.9858	1	[0004208, 0005432, 0007652]	342117.9	2032.3	1700
4	0.9841	0.7761	0.9774	0.9779	0.9825	[0001744, 0001531, 0007518, 0005346]	384991.6	2110.6	7600
5	0.9632	0.6983	0.9709	0.9712	0.9789	[0002792, 0000567, 0006014, 0003580, 0008069]	360911.7	2212.7	Unable*
6	0.9535	0.6621	0.9667	0.9676	0.9748	[0002899, 0003580, 0003876, 0004847, 0006014, 0009046]	329938.6	2166.4	Unable

* The program has been running for about four days on a system with Core i5 CPU, 2.27 GHz and 4.00 GB RAM on MATLAB environment, no solution has been found in this period of time.

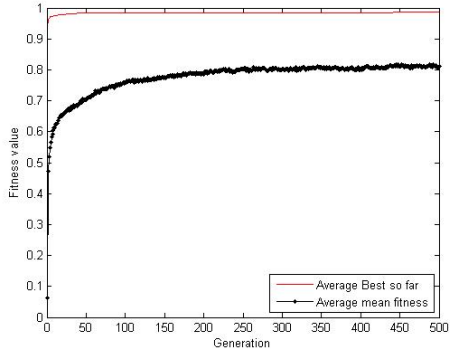

Fig. 4 The 3-similarity result on Re0 dataset

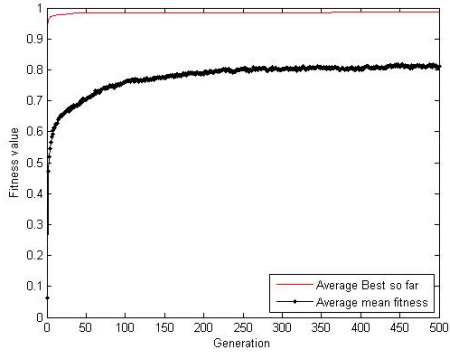

Fig. 5 The 4-similarity result on Re0 dataset

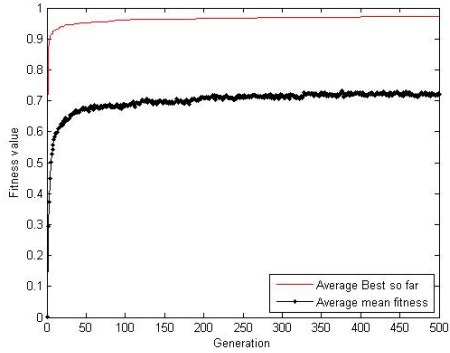

Fig. 6 The 5-similarity result on Re0 dataset

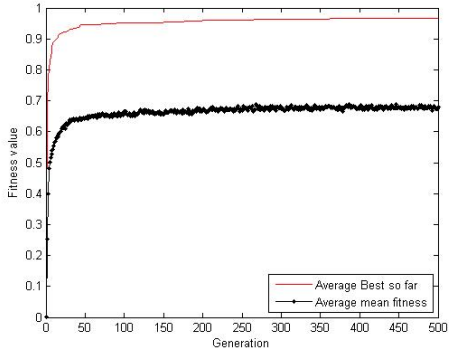

Fig. 7 The 6-similarity result on Re0 dataset

6. Concluding Remarks

The complexity of the n-similarity problem solved by GA is of order $ITER \times N \times n^2 = O(n^2)$, where $ITER$ and N are the number of iterations and the size of the population fixed and n is the size of desirable cluster. However the order of method in [23] for finding maximum n-similar cluster of m objects is computed as $m^3 + m^4 + \ + m^n = O(m^n)$. This states that the order of used method in [23] strictly depends on the size of dataset. It can be concluded that the method in [23] is more suitable for small size datasets. But in the case of large dataset situations, our approach by GA will be more efficient with less cost. The drawback of GA is that λ-cut set may not contain all possible solutions completely. This problem can relate to the random nature of GAs. In order to improve the solution set more, increasing the iterations number or the size of population are suggested in the algorithm. This issue is one of our directions for future works.

Acknowledgments

This article was extracted from the thesis prepared by Mina Mirhosseini to fulfil the requirements for earning the Master degree. I would like to express my sincere thanks to dear reviewers and editors for their guidance and valuable comments.

References

[1] M. Keshavarzi, M.A. Dehghan, M. Mashinchi, Applications of classification based on similarities and dissimilarities, Fuzzy Information and Engineering 4 (2012) 75-92.

[2] M. Keshavarzi, Classification based on similarity and dissimilarity, School of Mathematics and Computer Science, Shahid Bahonar University of Kerman, Iran, 2010.

[3] A. Tversky, Features of similarity, Psychological Review 84 (1977) 327-352.

[4] P. Resnik, Disambiguating noun groupings with respect to WordNet senses, Natural Language Processing Using Very Large Corpora Text, Speech and Language Technology 11 (1999) 77-98.

[5] D. Hindle, Noun classification from predicate-argument structures, Proceedings of the 28th Annual Meeting on Association for Computational Linguistics (1990) 268-275.

[6] W.B. Frakes, R. Baeza-Yates Information Retrieval: Data Structure and Algorithms, Prentice Hall, New Jersey, 1992.
[7] J.H. Lee, M.H. Kim, Y.J. Lee, Information retrieval based on conceptual distance in IS-A hierarchies, Journal of Documentation 49 (2) (1993) 188-207.
[8] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, 2008.
[9] M.J. McGill, M. Koll, T. Noreault, An Evaluation of Factors Affecting Document Ranking by Information Retrieval Systems, School of Information Studies, Syracuse University, United States, 1979.
[10] S.M. Chen, M.S. Yeh, P.Y. Hsiao A comparison of similarity measures of fuzzy values, Fuzzy Sets and Systems 72 (1995) 79-89.
[11] H. Rezaei, M. Emoto, M. Mukaidono, New similarity measure between two fuzzy sets, Advanced Computational Intelligence and Intelligent Informatics 10 (2006) 946-953.
[12] W.J. Wang, New similarity measures on fuzzy sets and on elements, Fuzzy Sets and Systems 85 (1997) 305-309.
[13] J. Ye, Cosine similarity measures for intuitionistic fuzzy sets and their applications, Mathematical and Computer Modeling 53 (2011) 91-97.
[14] G.J. Torres, R.B. Basnet, A.H. Sung, S. Mukkamala, B.M. Ribeiro, A similarity measure for clustering and its applications, Proceedings of World Academy of Science, Engineering and Technology 31 (2008) 490-496.
[15] R. Sarac, K. Tu, N.A. Allahverdi, Fuzzy clustering approach for finding similar documents using a novel similarity measure, Expert Systems with Applications 33 (2007) 600-605.
[16] M.W. Berry, M. Browne, Lecture Notes in Data Mining, World Scientific Publishing of Hackensack, New Jersey, 2006.
[17] A. Huang, Similarity measures for text document clustering, 6th New Zealand Computer Science Research Student Conference (2008) 49-56.
[18] N. Sandhya, Y.S. Lalitha, A. Govardhan, K. Anuradha, Analysis of similarity measures for text clustering, CSC Journals 2 (2008).
[19] M. Rafi, S.M. Shaikh, An improved semantic similarity measure for document clustering based on topic maps, CoRR, 2013.
[20] T.H. Cormen, E.C. Leiserson, L.R. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, Second Edition, 2001.
[21] G. Kondrak, N-gram similarity and distance, String Processing and Information Retrieval, Lecture Notes in Computer Science 3772 (2005) 115-126.
[22] S. Kakar, Identification of level of resemblance between web based documents, International Journal of Engineering and Computer Science 2 (2013) 3097-3100.
[23] M. Keshavarzi, M.A. Dehghan, M. Mashinchi, Classification based on 3-similarity, Iranian Journal of Mathematical Sciences and Informatics 6 (2011) 7-21.
[24] M. Melanie, An Introduction to Genetic Algorithm, Cambridge, MIT Press, 1996.
[25] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, United States, 1989.
[26] T. Weise, Global Optimization Algorithms Theory and Application, Thomas Wesley, 2009.
[27] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer-Verglag, New York, 2007.
[28] D.D. Lewis, Reuters-21578 text categorization test collection distribution1.0, http://www.research.att.com/ lewis/reuters21578.html, 1999.
[29] M.F. Porter, An algorithm for suffix stripping, Program 14 (1980) 130-137.