Experimental Comparison of Iterative Versus Evolutionary Crisp and Rough Clustering

Pawan Lingras [1] **Manish Joshi** [2]

[1] *Department of Mathematics and Computing Science, Saint Mary's University, Halifax, Nova Scotia, B3H 3C3, Canada.*

E-mail: pawan@cs.smu.ca

[2] *Department of Computer Science, North Maharashtra University, Jalgaon, Maharashtra, 425 001, India*

E-mail: joshmanish@gmail.com

Accepted: 19-11-2009
Received: 20-09-2010

Abstract

Researchers have proposed several Genetic Algorithm (GA) based crisp clustering algorithms. Rough clustering based on Genetic Algorithms, Kohonen Self-Organizing Maps, K-means algorithm are also reported in literature. Recently, researchers have combined GAs with iterative rough clustering algorithms such as K-means and K-Medoids. Use of GAs makes it possible to specify explicit optimization of cluster validity measures. However, it can result in additional computing time. In this paper we compare results obtained using K-means, GA K-means, rough K-means, GA rough K-means and GA rough K-medoid algorithms. We experimented with a synthetic data set, a real world data set, and a standard dataset using a total within cluster variation, average precision, and execution time required as the criteria for comparison.

Keywords: Rough Clustering, Crisp Clustering, GA based Clustering, Cluster Quality.

1. Introduction

Grouping of large number of objects into a smaller number of manageable groups makes it easier to formulate planning strategies. The grouping is usually based on a similarity measure. For example, customers to a supermarket can be grouped based on their buying patterns. The store can then design marketing campaigns for various groups, instead of individual customers. Two of the most well-known and commonly used partitioning methods are K-means and K-medoids. Given D, a data set of n objects, and k, the number of clusters to form, K-means and K-medoid algorithms organize the objects into k partitions ($k \leqslant n$), where each partition represents a cluster. The K-means as well as the K-medoid clustering partition the input data set into k clusters to optimize an objective partitioning criterion, such as dissimilarity function based on distance. A minimized total within cluster variation ensures that the objects within a cluster are 'similar'. However, the objective partitioning criterion is not explicitly optimized. Another drawback of K-means and K-medoid clustering is that it can fall in local optima.

Deterministic local search always converges to the nearest local optimum from a starting position of

```
Input:
    $k$: the number of clusters,
    $D(n,m)$: a data set containing $n$ objects where each object has $m$ dimensions,
    $\delta$:an input parameter that stands for a small acceptable change in the subsequent centroid values,
    $iter$: an input parameter indicating number of consecutive iterations for which
      difference in subsequent centroid values should be less than $\delta$,
Output:
    A set of clusters.
Steps:
    arbitrarily choose $k$ objects from $D$ as the initial cluster centers (centroids);
    repeat
        (re)assign each object to the cluster based on the distance of an
            object from the centroid of the cluster;
        update the cluster centroids using the number of objects assigned to each cluster;
    until no change;
```

Figure 1: The K-means algorithm for crisp clustering

the search, hence K-means result largely depends on the initial cluster centers. On the contrary, stochastic search heuristics inspired by evolution and genetics has the ability to cope with local optima by maintaining, recombining and comparing several candidate solutions simultaneously. GA's are believed to be effective on global optimization problems, and they can provide good near-optimal solutions in reasonable time. Hence, to minimize a total within cluster variation and overcome the trap of falling into local optima, use of genetic algorithms (GA's) is proposed. [1,2,3,4,5]

The conventional clustering categorizes an object into precisely one cluster. Whereas, fuzzy clustering[6,7,8] and rough set clustering [9,10,11,12,13,14] provide ability to specify the membership of an object to multiple clusters, which can be useful in real world applications. Clustering in relation to rough set theory is attracting increasing interest among researchers.[13,14,15,16,17,18] Lingras[19] described how a rough set theoretic clustering scheme can be represented using a rough set genome. Rough set genomes were used to find an optimal balance between rough within-group-error and precision. However, the space requirement for rough set genomes as well as the convergence of the evolutionary process can be an issue for a large dataset. In subsequent publications,[10,20] modifications of K-means and Kohonen Self-Organizing Maps (SOM) were proposed to create intervals of clusters based on rough set theory. Rough K-means algorithm and its variations[13,16] have been most popular methods for rough set clustering due to their simplicity and efficiency. However, rough K-means has not been shown to explicitly find an optimal clustering scheme for a particular cluster quality measure.

In this paper we discuss various aspects of crisp, rough set based, and evolutionary clustering algorithms. We discuss and present appropriate modifications required to the basic K-means and K-medoid algorithms so that these algorithms adapt to rough and evolutionary clustering. In particular, we explain K-means, GA K-means, rough K-means, GA rough K-means,[12] and GA rough K-medoid[21] algorithms and their corresponding fitness functions.

Section 2 describes a K-means crisp clustering algorithm. Section 3 describes how rough set theory is embedded with a K-means algorithm. Section 4 discusses the inclusion of GA to K-means, rough K-means, and rough K-medoid algorithms. The genome sizes of the rough K-means and rough K-medoids are compared favorably with the original rough set genome. We also discuss the formulation of a fitness function for GA based crisp and rough clustering in this section. The discussion of applying these algorithms to a synthetic data set, a real world library data set, a standard data set and analysis of resulting cluster quality thereof is presented in section 5, 6 and 7 respectively. Conclusions are reported in section 8.

2. Crisp Clustering

The conventional clustering techniques group objects into separate clusters. Each object is assigned to only one cluster. The term crisp clustering refers to the fact that the cluster boundaries are strictly defined and object's cluster membership is unambiguous.

2.1. Partitioning Algorithms

Algorithms of this category identify homogeneous groups of objects by finding similarities among objects. Objects are represented by their characterizing attributes. Clustering is considered as an optimization problem for which the objective is to maximize the similarities among objects within the same clusters while minimizing the dissimilarities between different clusters.

A clustering algorithm is desired to be simple, efficient and should deal with huge datasets. It should be able to detect different cluster shapes. Even though K-means is one of the popular partition clustering algorithms, it converges to arbitrary local optima and can not deal well with non-spherical shaped clusters[22]. We describe procedural steps for the K-means algorithms.

K-means Algorithm. K-means clustering is one of the most popular statistical clustering techniques.[23,24] The K-means algorithm partitions the objects into clusters so that the resulting intra-cluster similarity is high but the inter-cluster similarity is low. For this algorithm, the cluster similarity is measured using the mean value of the objects in a cluster. This mean value is called the cluster centroid. Fig. 1 summarizes the K-means procedure.

We modeled the 'until no change condition' of the algorithm (Fig. 1) using two parameters δ and $iter$. The minimum accepted variance among all centroid values for $iter$ number of consecutive iterations is specified by the value of δ. If $Centroid_t(c_i)$ gives a centroid value of a cluster i of the t^{th} iteration, then for consecutive $iter$ number of iterations $Centroid_t(c_i) - Centroid_{t-1}(c_i) \leqslant \delta$, where $1 \leqslant i \leqslant k$ and $t > 1$. We have set the values of δ and $iter$ to 0.0001 and 50 respectively.

The objects are iteratively assigned to appropriate clusters based upon its distance from the centroids of the clusters. For an object v , let $d(v, c_i)$ be the distance between itself and the centroid of cluster. An object is assigned to a cluster if it follows the condition $d(v, c_j) = min_{1 \leqslant i \leqslant k} d(v, c_i)$, where distance between a vector and a cluster centroid is given by:

$$d(v, c_i) \; = \; \sqrt{\sum_{j=1}^{m} (v_j - c_{i_j})^2}. \qquad (1)$$

It is possible that K-means algorithm may fall in local optima. Moreover, K-means algorithm can not assign an object to multiple clusters. Hence, in order to overcome the problem of local optima and to manage the multiple membership issue, Genetic Algorithms (GA) and rough set theory are combined with K-means algorithm. In the next section, we will first discuss the rough set theory based extensions of the crisp clustering using an iterative rough K-means algorithm. Subsequent sections will present GA based enhancements.

3. Rough Clustering

In addition to clearly identifiable groups of objects, it is possible that a data set may consist of several objects that lie on the fringes. The conventional clustering techniques mandate that such objects belong to precisely one cluster. Such a requirement is found to be too restrictive in many data mining applications[25]. In practice, an object may display characteristics of different clusters. In such cases, an object should belong to more than one cluster, and as a result, cluster boundaries necessarily overlap. Fuzzy set representation of clusters, using algorithms such as fuzzy C-means, makes it possible

```
Input:
    $k$: the number of clusters,
    $D(n,m)$: a data set containing n objects where each object has m dimensions,
    $p$: a threshold value (1.4),
    $w\_lower$: relative importance assigned to lower bound (0.75),
    $w\_upper$: relative importance assigned to upper bound (0.25),
    $\delta$, $iter$: as described in Fig. 1
Output:
    A set of clusters. Each cluster is represented by the objects in the lower region
    and in boundary region (upper bound)
Steps:
    arbitrarily choose $k$ objects from $D$ as the initial cluster centers (centroids);
    repeat
        (re)assign each object to lower/upper bounds of appropriate clusters by
            determining its distance from each cluster centroid,
        update the cluster means (centroids) using the number of objects assigned
            and relative importance assigned to lower bound and upper bound of the cluster;
    until no change;
```

Figure 2: The K-means algorithm for rough clustering

for an object to belong to multiple clusters with a degree of membership between 0 and 1[7]. In some cases, the fuzzy degree of membership may be too descriptive for interpreting clustering results. Rough set based clustering provides a solution that is less restrictive than conventional clustering and less descriptive than fuzzy clustering.

Lingras and West[10] provided an efficient alternative based on an extension of the K-means algorithm.[23,24] Incorporating rough sets into K-means clustering requires the addition of the concept of lower and upper bounds. The incorporation required redefinition of the calculation of the centroids to include the effects of lower and upper bounds. The next step was to design criteria to determine whether an object belonged to the lower and upper bounds of a cluster.

The rough K-means approach has been a subject of further research. Peters[13] discussed various refinements of Lingras and West's original proposal[10]. These included calculation of rough centroids and the use of ratios of distances as opposed to differences between distances similar to those used in the rough set based Kohonen algorithm described in[20]. The rough K-means[10] and its various extensions[12,13] have been found to be effective in distance based clustering. However, there is no theoretical work that proves that rough K-means explicitly finds an optimal clustering scheme. Moreover, the quality of clustering that is maximized by the rough clustering is not precisely defined. We compare crisp and rough clustering algorithm results.

Rough K-means Algorithm. We [26] represent each cluster $c_i, 1 \leqslant i \leqslant k$, using its lower $\underline{A}(c_i)$ and upper $\overline{A}(c_i)$ bounds. All objects that are clustered using the algorithm follow basic properties of rough set theory such as:

(P1) An object $\vec{x}$ can be part of at most one lower bound

(P2) $\vec{x} \in \underline{A}(\vec{c}_i) \Longrightarrow \vec{x} \in \overline{A}(\vec{c}_i)$

(P3) An object $\vec{x}$ is not part of any lower bound

$$\Updownarrow$$

$\vec{x}$ belongs to two or more upper bounds.

Fig. 2 depicts the general idea of the algorithm.

An object is assigned to lower and/or upper bound of one or more clusters. For each object vector, $\vec{v}$, let $d(\vec{v},\vec{c}_j)$ be the distance between itself and the centroid of cluster $\vec{c}_j$. Let $d(\vec{v},\vec{c}_i) = \min_{1 \leqslant j \leqslant k} d(\vec{v},\vec{c}_j)$. The ratios $d(\vec{v},\vec{c}_i)/d(\vec{v},\vec{c}_j)$, $1 \leqslant i,j \leqslant k$, are used to determine the membership of $\vec{v}$. Let $T = \{j : d(\vec{v},\vec{c}_i)/d(\vec{v},\vec{c}_j) \leqslant \mathit{threshold} \text{ and } i \neq j\}$.

1. If $T \neq \emptyset$, $\vec{v} \in \overline{A}(\vec{c}_i)$ and $\vec{v} \in \overline{A}(\vec{c}_j), \forall j \in T$. Furthermore, $\vec{v}$ is not part of any lower bound. The above criterion guarantees that property (P3) is satisfied.

2. Otherwise, if $T = \emptyset$, $\vec{v} \in \underline{A}(\vec{c}_i)$. In addition, by property (P2), $\vec{v} \in \overline{A}(\vec{c}_i)$.

It should be emphasized that the approximation space A is not defined based on any predefined relation on the set of objects. The lower and upper bounds are constructed based on the criteria described above.

The values of p (a threshold), *w_lower*, *w_upper* are finalized based on the experiments described in[27].

4. Evolutionary Clustering Algorithms

This section contains some of the basic concepts of genetic algorithms as described in[1]. A genetic algorithm is a search process that follows the principles of evolution through natural selection. The domain knowledge is represented using a candidate solution called an *organism*. Typically, an organism is a single *genome* represented as a vector of length n:

$$g = (g_i \mid 1 \leqslant i \leqslant n), \tag{2}$$

where g_i is called a *gene*.

A group of organisms is called a *population*. Successive populations are called *generations*. A generational GA starts from initial generation $G(0)$, and for each generation $G(t)$ generates a new generation $G(t+1)$ using genetic operators such as *mutation* and *crossover*. The mutation operator creates new genomes by changing values of one or more genes at random. The crossover operator joins segments of two or more genomes to generate a new genome.

Fig. 3 describes the evolutionary procedure. The evaluation process of a genome i.e. evaluate $G(t)$, is a combination of two steps. The first step determines membership of all objects to corresponding clusters. As described in earlier section, appropriate calculations for crisp and rough clustering figures out the members of each cluster. In the next step, fitness of the genome is determined. The intuitive distance measure is used to decide the fitness of the genome. There is an obvious difference between the fitness calculations for crisp clustering and rough clustering. The fitness formulas used for both clustering are described below.

Genome Fitness Function for Crisp Clustering. The fitness is calculated based on the allocation of all objects to the clusters. It is given by:

$$\mathit{Fitness} = \sum_{i=1}^{k} \sum_{u \in c_i} d(u,x_i). \tag{3}$$

Fitness is the sum of the Euclidean distances for all objects in the cluster; u is the point in space representing a given object; and x_i is the centroid/medoid of cluster c_i (both u and x_i are multidimensional). The function d provides the distance between two vectors. The distance $d(u,x)$ is given by:

$$d(u,x) = \sqrt{\sum_{j=1}^{m} (u_j - x_j)^2}. \tag{4}$$

Here, the value of m indicates the total number of dimensions.

Genome Fitness Function for Rough Clustering. The Fitness function has to change to adapt to the rough set theory by creating lower and upper versions of the Fitness as:

$$\underline{\mathit{Fitness}} = \sum_{i=1}^{k} \sum_{u \in \underline{A}(c_i)} d(u,x_i), \tag{5}$$

$$\overline{\mathit{Fitness}} = \sum_{i=1}^{k} \sum_{u \in \overline{A}(c_i)} d(u,x_i), \tag{6}$$

where $\underline{A}(c_i)$ and $\overline{A}(c_i)$ represents lower and upper bound of cluster c_i. The distance function d does not

```
Input:
        All the parameters present in Fig 1 (for crisp clustering) and
          Fig. 2(for rough clustering)
        population: The number of organisms to be generated,
        generations: The number of successive populations to be generated,
Output:
        A set of clusters. Each cluster is represented by the objects in the lower region
        and in boundary region (upper bound)
Steps:
        generate initial population, $G(0)$;
        evaluate $G(0)$;
        for ($t$=1; $t \leqslant generations$; $t++$)
                generate $G(t)$ using $G(t-1)$;
                evaluate $G(t)$;
```

Figure 3: Steps used in the GA based Evolutionary algorithms

change. The *Fitness* value for the rough clustering is calculated as

$$Fitness = w\_lower \times \underline{Fitness} + w\_upper \times \overline{Fitness}. \quad (7)$$

where *w_lower* and *w_upper* are relative importance assigned to lower and upper bound of the clusters.

Thus evolutionary algorithms for clustering differ mostly in the genome representation and the fitness calculation. The variations of GA based algorithms for crisp and rough clustering are described in next subsections. All these GA based algorithms follow steps depicted in Fig. 3.

4.1. GA K-means

The K-means algorithm discussed in section 2 is modified to adapt the principles of GA. The organism has a total of $k \times m$ genes. A batch of every m genes represents centroids of corresponding clusters. The population size and generation values are input parameters.

4.2. Rough set genome

Before studying the GA Rough K-means, we will look at the first attempt at rough clustering using GAs proposed by Lingras[19] in 2001 using rough set genome.

A rough set genome consists of n genes, one gene per object in U. A gene for an object is a string of bits that describes which lower and upper approximations the object belongs to. Properties (P1)-(P3) provide certain restrictions on the memberships. An object $\mathbf{u} \in U$ can belong to the lower approximation of at most one class $\mathbf{x}_i$. If an object belongs to the lower approximation of $\mathbf{x}_i$ then it also belongs to the upper approximation of $\mathbf{x}_i$. If an object does not belong to the lower approximation of any $\mathbf{x}_i$, then it belongs to the upper approximation of at least two (possibly more) $\mathbf{x}_i$.

Based on these observations the string for a gene can be partitioned into two parts, *lower* and *upper*. Both the lower and upper parts of the string consist of k bits each. The i^{th} bit in lower/upper string tells whether the object is in the lower/upper approximation of $\mathbf{x}_i$.

If $\mathbf{u} \in \underline{A}(\mathbf{x}_i)$, then based on the property (P2), $\mathbf{u} \in \overline{A}(\mathbf{x}_i)$. Therefore, the i^{th} bit in both the lower and upper strings will be turned on. Based on the property (P1) all the other bits must be turned off.

If $\mathbf{u}$ is not in any of the lower approximations, then according to property (P3), it must be in two or more upper approximations of $\mathbf{x}_i, 1 \leqslant i \leqslant k$, and cor-

Valid genes

	Lower			Upper		
	$\underline{A}(\mathbf{x}_3)$	$\underline{A}(\mathbf{x}_2)$	$\underline{A}(\mathbf{x}_1)$	$\overline{A}(\mathbf{x}_3)$	$\overline{A}(\mathbf{x}_2)$	$\overline{A}(\mathbf{x}_1)$
$gene_1$	0	0	0	0	1	1
$gene_2$	0	0	0	1	0	1
$gene_3$	0	0	0	1	1	0
$gene_4$	0	0	0	1	1	1
$gene_5$	0	0	1	0	0	1
$gene_6$	0	1	0	0	1	0
$gene_7$	1	0	0	1	0	0

Some examples of invalid genes

	Lower			Upper		
	$\underline{A}(\mathbf{x}_3)$	$\underline{A}(\mathbf{x}_2)$	$\underline{A}(\mathbf{x}_1)$	$\overline{A}(\mathbf{x}_3)$	$\overline{A}(\mathbf{x}_2)$	$\overline{A}(\mathbf{x}_1)$
$invalidGene_1$	0	0	1	0	0	0
$invalidGene_2$	0	1	0	1	1	0
$invalidGene_3$	1	0	1	0	0	0
$invalidGene_4$	0	0	0	0	0	1

Figure 4: Genes in a rough set genome

responding i^{th} bits in the upper string will be turned on.

Fig. 4 shows examples of all the valid and some of the invalid genes for $k = 3$. Genes $gene_1$ to $gene_7$ are all the acceptable values of genes for $k = 3$. An object represented by $gene_1$ belongs to $\overline{A}(\mathbf{x}_1)$ and $\overline{A}(\mathbf{x}_2)$. An object represented by $gene_6$ belongs to $\underline{A}(\mathbf{x}_2)$, and by property (P2) to $\overline{A}(\mathbf{x}_2)$.

Any other value not given by $gene_1$ to $gene_7$ is not valid. Fig. 4 also shows four of the 57 invalid values. The $invalidGene_1$ is invalid because an object cannot be in $\underline{A}(\mathbf{x}_1)$ and not be in $\overline{A}(\mathbf{x}_1)$. The $invalidGene_2$ is invalid because an object cannot be in $\underline{A}(\mathbf{x}_2)$ and in $\overline{A}(\mathbf{x}_3)$ at the same time. The $invalidGene_3$ is invalid because an object cannot be in $\underline{A}(\mathbf{x}_1)$ and in $\underline{A}(\mathbf{x}_3)$ at the same time. Since the object represented by $invalidGene_4$ only belongs to $\overline{A}(\mathbf{x}_1)$, according to property (P3) it is invalid.

A genetic algorithm package such as the one used in the study[28] makes it possible to describe a set of valid gene values or alleles. All the standard genetic operations will then only create genomes that have these values. Therefore, the conventional genetic operations can be used with rough set genomes in such a package. Lingras[19] evolved a rough set clustering by optimizing the cluster validity measure such as the one given by "Eq. 7". One of the major drawbacks of the rough set genome was the fact that the size of the genome was directly proportional to the number of objects. Therefore, the approach was feasible for a relatively small number of objects. The following sections describe two approaches that are based on the rough K-means algorithm that progressively reduce the size of the genome.

4.3. GA Rough K-means

Mitra[12] proposed an evolutionary rough clustering algorithm that used the genomes whose size was proportional to the number of clusters, k. Since the number of clusters k are significantly smaller than the number of objects n, it is possible to apply the algorithm to large datasets. In this section, we present a similar approach described in[26], which is essentially an evolutionary modification of the rough K-means algorithm. The objective of the approach was to explicitly evolve an optimal clustering scheme. The proposed genome for the evolutionary algorithm has a total of $k \times m$ genes, where k is the desired number of clusters and m is the number of dimensions used to represent objects and centroids. The first m genes represent the first centroid. Genes $m+1, \ldots, 2 \times m$ give us the second centroid, and so on. Finally, $((k-1) \times m) + 1, \ldots, k \times m$ corresponds to the k^{th} centroid. We apply this genome configuration in an algorithm that is used for comparison. The cluster assignment is similar to rough K-means defined in Fig. 2. The rough fitness measure given by "Eq. 7" is minimized.

4.4. GA Rough K-medoid

The size of the genome used for rough clustering can be further reduced with the help of a modified K-medoid algorithm as was proposed by Peters, et al.[13] Unlike K-means algorithm where mean value is used as a centroid of a cluster, in K-medoid algorithm actual object is used as a reference point of a cluster. One object is used to represent a cluster and remaining objects clustered based on their similarity with the representative object.

A medoid is the most centrally located object in a given cluster. For k clusters, we will have k medoids. A genetic algorithm can be used to search for the most appropriate k medoids. The genome will contain k genes, each corresponding to a medoid. This reduces the size of a genome from $k \times m$ by a factor of m to k. The steps in this algorithm are similar to that of GA rough K-means. The major difference is the use of medoids instead of centroids of the clusters.

Lingras[29] describe theoretical and experimental comparison between GA based rough K-means and rough K-medoids algorithms. The gene values for rough K-medoids are discrete values corresponding to object IDs as opposed to continuous real variables used for centroids in the rough K-means evolution. This results in a restricted search space for the proposed rough K-medoids leading to further increase in the chances of convergence. This paper provides a more elaborate experimental testing of this hypothesis.

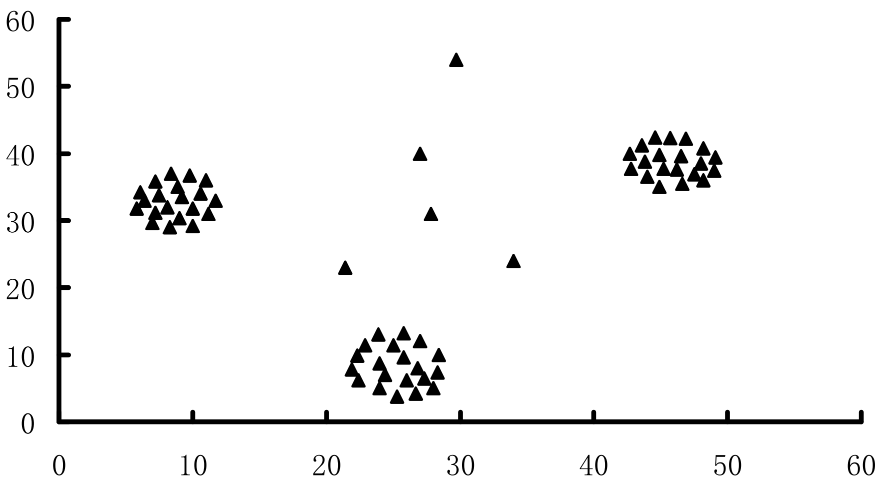

Figure 5: Synthetic data

5. Comparative Results - Synthetic Data set

We used the synthetic data set developed by Lingras et al.[30] to test the validity of the evolutionary rough K-means. There are a total of 65 objects as shown in Fig. 5. It is obvious that there are three distinct clusters. However, five objects do not belong to any particular cluster.

Since there are three distinct clusters in the data set, the number of clusters for all the algorithms was set to three. The GAs used the crossover probability of 70% and mutation probability of 10%. For each algorithm, best and average Fitness for five trials are presented. We carried experiments with different population sizes and generations. Table 1 shows the results obtained using K-means and rough K-means algorithms. We can see that clusters generated by rough K-means have less intra-cluster variation. That means for the synthetic data set, the rough K-means provides better cluster quality as compared to the crisp K-means algorithm.

Table 1 also shows the number of generations and population sizes required to approach or exceed the cluster quality obtained from the iterative algorithms. The table compares K-means against GA K-means and rough K-means against GA rough K-means.

The average K-means results are improved by GA K-means. The best result obtained using K-means (47.0268) is not reached by GA K-means even for 500 population size and 500 generations (47.0269). But due to the problem of local optima, K-means could not generate good results consistently. Whereas the GAs results with adequate numbers of populations and generations (in this case 100 each) are consistent, which leads to improved performance. For lesser number of objects (65) the time required for the GAs is not significant, but for more number of objects it would be interesting to see the execution times for GAs.

We can see that the rough K-means result is also improved using GA rough K-means. Big variations between the best and average Fitness values for rough K-means and GA rough K-means indicate that sometimes both algorithms are stuck in local optima. Appropriate increase in population size and generations solves this problem for GA (in this case 100 each).

Table 2 shows comparison of evolutionary rough clustering algorithms against the basic K-means algorithm. Due to the use of medoid (one of the objects) rather than centroid (any point in space) local optima does not pose a major problem. Hence, GA rough K-medoid results are more consistent than GA K-means and GA rough K-means algorithms. GA rough K-means requires population size of 100

Table 1: Comparison of crisp, rough and evolutionary algorithms for Synthetic data.

K-means Fitness	Population size Generations	GA K-means Fitness (Best/Avg)	GA K-means Average Time (Sec)	Rough K-means Fitness	GA Rough K-means Fitness (Best/Avg)	GA Rough K-means Average Time (Sec)
47.03 / 63.34	10,20	132.17 / 163.79	< 1	45.19 / 53.84	97.02 / 124.56	< 1
	20,20	110.81 / 123.29	< 1		94.36 / 107.36	< 1
	20,30	68.56 / 90.43	< 1		70.03 / 104.33	< 1
	30,50	63.10 / 92.90	< 1		89.43 / 90.57	< 1
	50,50	54.48 / 73.24	< 1		41.36 / 78.45	< 1
	100,100	**47.06 / 47.19**	**1**		**35.04 / 35.29**	**1**

Table 2: Comparison of K-means, GA rough K-means and GA Rough K-medoid algorithms.

K-means Fitness	Population size Generations	GA Rough K-means Average Fitness	GA Rough K-means Average Time (Sec)	GA Rough K-medoid Average Fitness	GA Rough K-medoid Average Time (Sec)
47.03 / 63.34	10,20	97.02 / 124.56	< 1	**36.83 / 55.32**	< 1
	20,20	94.36 / 107.36	< 1	36.40 / 37.33	< 1
	20,30	70.03 / 104.33	< 1	35.70 / 36.32	< 1
	30,50	89.43 / 90.57	< 1	36.24 / 36.33	< 1
	50,50	41.36 / 78.45	< 1	36.24 / 36.28	< 1
	100,100	**35.04 / 35.29** / 1	**1**	35.70 / 36.03	1

and 100 generations to match K-means, whereas GA rough K-medoid needs the population size of 10 and 20 generations. Faster convergence for GA rough K-medoid implies lesser time requirements. This time difference will be more significant for a larger data set.

After experimenting with the small data set we carried out experiments on a real world data and a standard data set. The details of the data sets and results are discussed in the next sections.

6. Comparative Results - Library Data set

We used the data obtained from a public library to compare the results of various algorithms. The data consist of books borrowed by members. The objective is to group members with similar reading habits. Information about how many times a member borrows books of a particular category is collected. The data is normalized in the range of 0 to 1 to reduce the effect of outliers. In order to visualize the data set, it is restricted to two dimensions. Data of 1895 members is presented as a scatter graph in Fig. 6. It shows propensity of a member to borrow a book of a certain category. There are no obvious clusters in the figure. It will be interesting to see how all the algorithms carve out reasonable clusters from such a scattered data.

The parameters of the crisp, rough and evolutionary algorithms are not changed. Table 3 shows the results obtained using K-means and rough K-means algorithms. Clusters generated by rough K-means have less intra-cluster variation than for the clusters generated by K-means algorithm. Hence we can conclude that the rough K-means results in better cluster quality than the crisp K-means algorithm. We have seen similar observations for the synthetic data set in the earlier section.

Table 3 also shows the comparison of K-means against GA K-means and rough K-means against GA rough K-means algorithms. The results include average Fitness from five trials. For a normal range of population size and generations the GA K-means does not outperform the K-means. But for population size of 500 and 500 generations the average Fitness of GA K-means for 3 clusters is less than K-means. This performance improvement requires 90 seconds of computation. For five clusters the GA K-means (population size of 500 and 500 generations) could outperform K-means at the cost of 111.2 seconds of processing time.

As the data set size increases the population size and generations of GA should be increased to obtain improved Fitness. In some cases, the higher computational cost of GA K-means may not be justified by slight increase in accuracy.

Table 3 shows that GA rough K-means improves the results of rough K-means. Moreover, GA rough K-means does this with far less population size and generations than GA K-means.

Likewise the synthetic data set, both GA rough K-medoid and GA rough K-means results are better than the K-means results. The GA rough K-medoid algorithm is faster than the GA rough K-means in surpassing the K-means results (see Table 4). GA rough K-means requires 4.6 seconds with 30 population size and 50 generations to get better solution than K-means. Whereas GA rough K-medoid with 20 population size and 20 generations surpasses the K-means results in 3 seconds.

Fig. 7 shows the performance of GAs for different configurations. GA K-means performs better than K-means for large number of population size and generations. GA rough K-medoid algorithm promptly outdoes the K-means results, but for higher population size and generations GA rough K-means generates optimal results.

7. Comparative Results - Standard Data set

We used Letter Recognition [31] data from the University of California Irvine machine learning repository. The data set contains 20,000 events representing character images of 26 capital letters in the English alphabet based on 20 different fonts. Each event is represented using 16 primitive numerical attributes (statistical moments and edge counts) that are scaled to fit into a range of integer values from 0 through 15.

In order to crosscheck the results and compare the cluster quality we decided to consider limited number of characters. We prepared two data sets. In

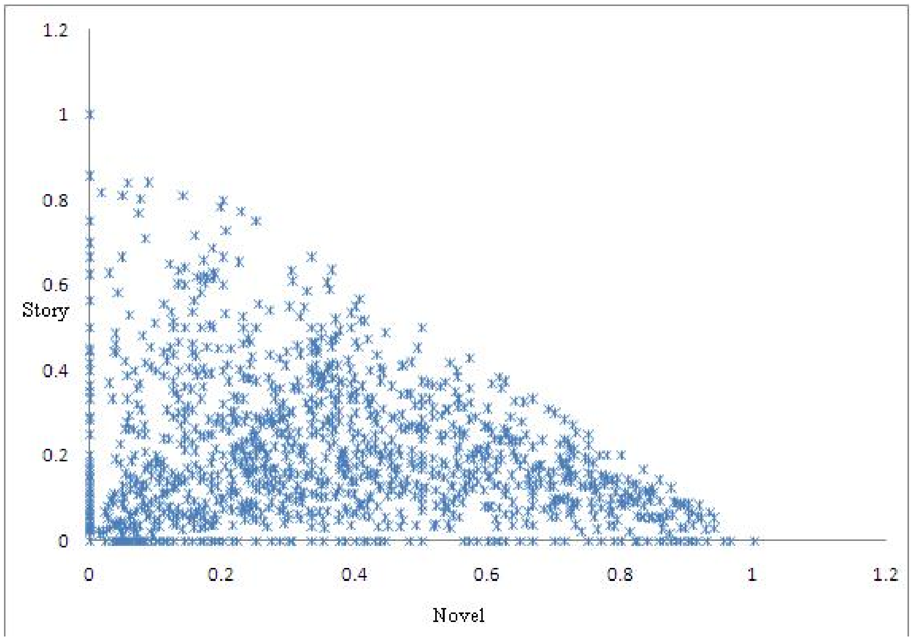

Figure 6: Library data

Table 3: Comparison between crisp and evolutionary algorithms for Library data.

Average K-means Fitness	Population size Generations	GA K-means Average fitness	GA K-means Average Time (Sec)	Average Rough K-means Fitness	GA Rough K-means Average fitness	GA Rough K-means Average Time (Sec)
3 Clusters						
7.9764	10,20	9.1223	1	6.9517	8.094	1.2
	20,20	8.6642	1		7.82	2.4
	20,30	8.3182	1.4		7.12	2.4
	30,50	8.0323	1.6		**6.91**	**4.4**
	50,50	8.0261	2.8		6.9001	6
	100,100	7.9780	6.8		6.9006	19
	500,500	7.9761	90		-	-
5 Clusters						
5.9020	10,20	8.9841	1	5.2061	7.1033	1.4
	20,20	6.7930	1.8		6.7505	2.8
	20,30	7.4541	1.8		6.2648	3
	30,50	6.3575	2.8		5.8794	4.6
	50,50	6.4889	3.2		5.2801	8
	100,100	5.9527	16.4		**5.0730**	**25.8**
	500,500	**5.9002**	**111.2**		-	-

Table 4: Comparison of K-means, GA rough K-means and GA Rough K-medoid for Library Data set.

K-means Fitness	Population size Generations	GA Rough K-means Average Fitness	GA Rough K-means Average Time (Sec)	GA Rough K-medoid Average Fitness	GA Rough K-medoid Average Time (Sec)
3 Clusters					
7.9764	10,20	8.094	1.2	**7.7042**	**1.2**
	20,20	7.82	2.4	7.1390	2.2
	20,30	7.12	2.4	7.2670	2.8
	30,50	**6.91**	**4.4**	7.0719	4
	50,50	6.9001	6	7.0154	6.4
	100,100	6.9006	19	7.0465	19
5 Clusters					
5.9020	10,20	7.1033	1.4	6.0141	2
	20,20	6.7505	2.8	**5.3114**	**3**
	20,30	6.2648	3	5.3502	3.2
	30,50	**5.8794**	**4.6**	5.1660	5
	50,50	5.2801	8	5.1770	8.6
	100,100	5.0730	25.8	5.1034	21.4

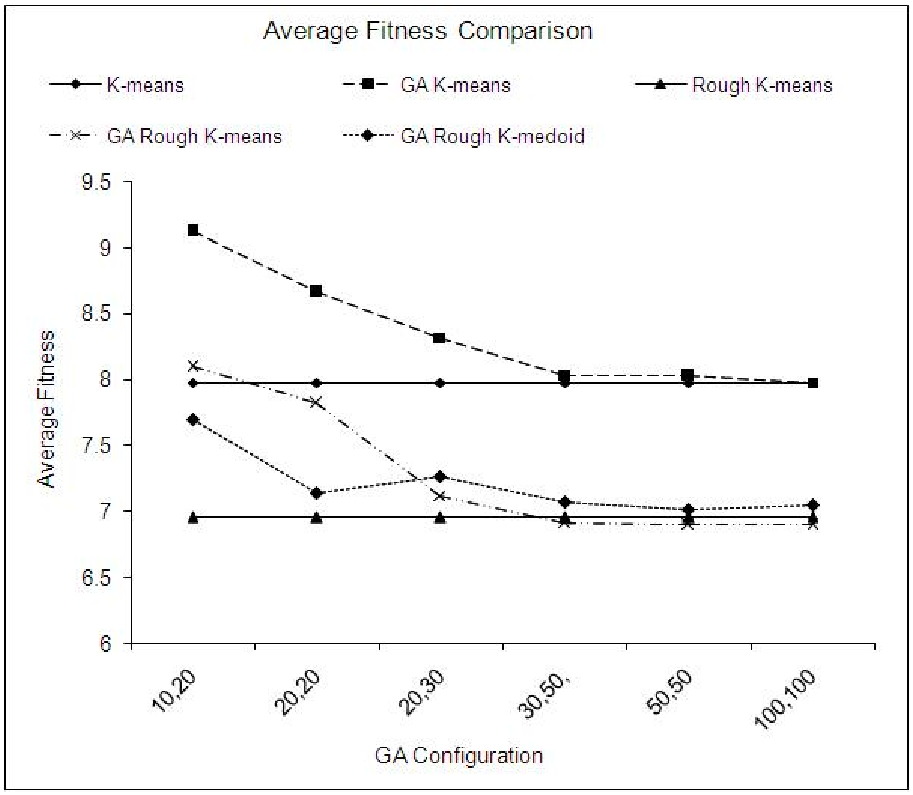

Figure 7: Comparison chart of different algorithms.

Table 5: K-means algorithm clustering for characters from A to G.

Cluster No.	Cluster Label Character	Frequency	Precision	Average Precision	Fitness
0	F	552/557	0.99		
1	D	362/1159	0.31		
2	D	309/829	0.37		
3	C	275/681	0.40	0.54	325.56
4	C	309/558	0.55		
5	A	688/704	0.98		
6	B	182/924	0.20		

Table 6: Rough K-means algorithm clustering for characters from A to G.

Cluster No.	Cluster Label Character	Frequency	Precision	Average Precision	Fitness
0	C	392/645	0.61		
1	A	198/208	0.95		
2	D	121/546	0.39		
3	F	494/515	0.96	0.65	296.57
4	B	262/727	0.36		
5	A	365/365	1.0		
6	G	120/479	0.25		

Table 7: K-means algorithm clustering for selected set of characters.

Cluster No.	Cluster Label Character	Frequency	Precision	Average Precision	Fitness
0	M	412/1119	0.37		
1	Z	276/880	0.31		
2	Z	378/734	0.51		
3	P	607/639	0.95	0.60	406.23
4	A	387/660	0.59		
5	A	311/514	0.61		
6	L	328/328	1.0		
7	O	600/1240	0.48		

Table 8: Rough K-means algorithm clustering for selected set of characters.

Cluster No.	Cluster Label Character	Frequency	Precision	Average Precision	Fitness
0	L	335/335	1.0	0.82	372.23
1	A	553/684	0.81		
2	Z	536/705	0.76		
3	A	183/255	0.72		
4	A	358/362	0.99		
5	S	96/294	0.33		
6	M	292/306	0.95		
7	P	553/555	1.0		

the first set, we kept 5412 events representing letter A to G. The second data set consists of 8 distinctly different characters A, H, L, M, O, P, S, and Z. This data set consists of 6114 events.

Besides using the Fitness measure for comparison we calculated the average precision for each cluster. Each cluster is labeled using the character that appears most frequently in the cluster. Precision of the cluster is defined as the number of events that match the cluster label divided by the total number of events in the cluster. Average precision is the average of all cluster precisions. Table 5 and Table 6 show the results obtained using K-means and rough K-means algorithm for the set 1.

The rough K-means algorithm not only outperforms the K-means algorithm in Fitness and in average precision, but also identify more letters correctly than K-means algorithm. Letter E is not prominently identified in any cluster generated by rough K-means whereas two letters E and G are not prominently identified by any cluster generated by K-means algorithm.

Table 7 and Table 8 show the comparison of results obtained using K-means and rough K-means algorithms for a set of selected characters. Since the selected characters are distinctly different, clustering quality ought to improve as compared to the first set of characters.

We can see that the average precision is improved for the set of selected characters. Rough K-means clustering generates better quality clusters than the crisp K-means algorithm. This conclusion is supported by reduced Fitness and increase in average precision. Moreover, only one letter 'H' is not prominently identified by rough K-means whereas two letters 'H' and 'S' are not prominently identified by any clusters generated using K-means algorithm.

GA based algorithms however generate poor results for this standard data set. For crisp as well as for rough clustering, most of the objects are grouped into few clusters. The remaining clusters are left empty. Further investigations are necessary to determine the reasons for failure of GAs for the character data set.

8. Conclusions

Iterative clustering algorithms tend to be efficient, but do not necessarily provide the optimal clustering. Moreover, it is not possible to specify a particular cluster validity measure as an optimization criterion in such algorithms. Genetic Algorithms, while computationally expensive, can be very effective in optimizing any given cluster validity measure.

The focus of this paper is to study the effectiveness of evolutionary rough clustering techniques. Rough clustering is more flexible than the conventional crisp clustering as it allows for an object to belong to more than one cluster with the help of boundary regions. One of the first approaches to rough clustering was based on genetic algorithms[19]. The size of rough set genome used in this original approach was directly proportional to the number of objects, *n*. As a result, the algorithm can only be applied to small datasets. Combining GAs with the object assignment used by a popular iterative algorithm called rough K-means, significantly reduces the size of the genome. The genome used in the evolutionary rough K-means algorithm only has $k \times m$ genes, where k is the number of clusters and m is the number of dimensions used to represent an object. Since $k \gg n$, for a reasonable value of m, the smaller genome reduces the memory requirement and increases the chances of evolving to a near optimal solution based on a specified criterion.

Use of medoids - most centrally located objects - to represent a cluster instead of centroid, would mean that we only need to keep track of k objects in a genome. As a result, evolutionary rough K-medoids can further reduce the size of the genome to just k. The gene values for rough K-medoids are discrete values corresponding to object IDs as opposed to continuous real variables used for centroids in the rough K-means evolution. This results in a restricted search space for the proposed rough K-medoids leading to further increase in the chances of convergence.

This paper further provided experimental comparison of results obtained by K-means, GA K-means, rough K-means, GA rough K-means and GA rough K-medoid algorithms. We applied all algorithms to a synthetic data set, a real world data set, and a standard data set. A simple and intuitive measure of total within cluster variation (Fitness) is used for the evaluation.

The rough K-means algorithm seems to provide better cluster quality in terms of the Fitness and average precision than the crisp K-means algorithm.

For sufficiently high population size and generations, GA K-means can improve average performance of K-means. As the size of data set increases, higher population size and generations are required in GA K-means algorithms to outperform the K-means results. Execution time increases when GAs with higher population size and generations are used. There is a trade off between execution time and better cluster quality when the GA K-means algorithm is used.

GA rough K-medoid converges faster and surpasses the K-means results with smaller populations and fewer generations than GA rough K-means. But for larger population and more generations the GA rough K-means results are superior to all other algorithms.

GA effect on rough clustering is more promising than that on crisp clustering. For both small as well as large data sets, the GA rough K-means and the GA rough K-medoid generate better clustering in reasonable amount of execution time.

References

1. B. P. Buckles and F. E. Petry, "Genetic Algorithms," *IEEE Computer Press*(1994).
2. S. Cheng, Y. Chao, H. Wang et al., "A Prototypes-Embedded Genetic K-Means Algorithm," *ICPR '06: Proc. IEEE Intl. Conf. on Pattern Recognition* , **2**, 724–727 (2006).
3. K. Krishna and M. Narasimha Murty, "Genetic K-Means Algorithm," *Systems, Man, and Cybernetics*, **29**, (1999).
4. Y. Lu, S. Lu, F. Fotouhi et al., "FGKA: A Fast Genetic K-Means Clustering Algorithm," *Proc. ACM Symposium on Applied Computing*, (2004).
5. U. Maulik and S. Bandyopadhyay, "Genetic Algorithm-Based Clustering Technique," *Pattern Recognition*, **33**, 1455–1465 (2000).
6. J. C. Bezdek and R. J. Hathaway, "Optimization of Fuzzy Clustering Criteria using Genetic Algorithms," (1994).
7. W. Pedrycz and J. Waletzky, "Fuzzy Clustering with Partial Supervision," *IEEE Trans. on Systems, Man and Cybernetics*, **27(5)**, 787–795 (1997).
8. S. Ilhan, N. Duru, E. Adali, "Improved Fuzzy Art Method for Initializing K-means," *International Journal of Computational Intelligence Systems*, **3(3)**, 274–279 (2010).
9. T. B. Ho and N. B. Nguyen, "Nonhierarchical Document Clustering by a Tolerance Rough Set Model," *International Journal of Intelligent Systems*, **17**, 199–212 (2002).
10. P. Lingras and C. West, "Interval Set Clustering of Web Users with Rough K-Means," *Journal of Intelligent Information Systems*, **23**, 5–16 (2004).
11. H. Bustince, "Interval-valued Fuzzy Sets in Soft Computing," *International Journal of Computational Intelligence Systems*, **3(2)**, 215–222 (2010).
12. S. Mitra, "An Evolutionary Rough Partitive Clustering," *Pattern Recognition Letters*, **25(12)**, 1439–1449 (2004)
13. G. Peters, "Some Refinements of Rough k-Means," *Pattern Recognition*, **39**, 1481–1491 (2006).
14. J. F. Peters, A. Skowron, Z. Suraj et al., "Clustering: A Rough Set Approach to Constructing Information Granules," *Soft Computing and Distributed Process-*

ing, 57–61 (2002).

15. S. Hirano and S. Tsumoto, "Rough Clustering and its Application to Medicine," *Journal of Information Science*, **124**, 125–137 (2000).
16. S. Mitra, H. Bank, and W. Pedrycz, "Rough-Fuzzy Collaborative Clustering," *IEEE Trans. on Systems, Man and Cybernetics*, **36(4)**, 795–805 (2006).
17. H. S. Nguyen, "Rough Document Clustering and the Internet," *Handbook on Granular Computing*, (2007).
18. K. E. Voges, N. K. Ll. Pope and M.R. Brown, "Cluster Analysis of Marketing Data: A Comparison of K-Means, Rough Set, and Rough Genetic Approaches," *Heuristics and Optimization for Knowledge Discovery*, Idea Group Publishing, 208–216 (2002).
19. P. Lingras, "Unsupervised Rough Set Classification using GAs," *Journal Of Intelligent Information Systems*, **16(3)**, 215–228, (2001).
20. P. Lingras, M. Hogo, M. Snorek, "Interval Set Clustering of Web Users using Modified Kohonen Self-Organizing Maps based on the Properties of Rough Sets," *Web Intelligence and Agent Systems: An International Journal*, **2(3)**, 217–230 (2004).
21. G. Peters, M. Lampart and R. Weber, "Evolutionary Rough k-Medoid Clustering," *Transactions on Rough Sets*, **VIII**, 289–306 (2008).
22. S. Paterlini and T. Krink, "High Performance Clustering with Differential Evolution," *Congress on Evolutionary Computations*, **2**, 2004–2011 (2004).
23. J. A. Hartigan and M.A. Wong, "Algorithm AS136: A K-Means Clustering Algorithm," *Applied Statistics*, **28**, 100–108 (1979).
24. J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, **1**, 281–297 (1967).
25. A. Joshi and R. Krishnapuram, "Robust Fuzzy Clustering Methods to Support Web Mining," *Proc. ACM SIGMOD Workshop Data Mining and Knowledge Discovery*, 1–8, (1998).
26. P. Lingras, "Evolutionary rough K-means Algorithm," *Proc. The Fourth International conference on Rough Set and Knowledge Technology (RSKT2009)* (2009).
27. P. Lingras, "Precision of rough set clustering," *LNCS: Rough Sets and Current Trends in Computing*, **5306/2008**, 369–378 (2008).
28. M. Wall and A. Galib, "C++ Library of Genetic Components," http://lancet.mit.edu/ga/, (1993).
29. P. Lingras, "Rough K-medoid Clustering using GAs," *Proc. of ICCI 2009*, (2009).
30. P. Lingras, M. Chen and D. Miao, "Rough Multi-category Decision Theoretic Framework," *Proc. of 3rd Intl. Conf. on Rough Sets and Knowledge Technology*, 676–683 (2008).
31. A. Asuncion and D. J. Newman, "UCI Machine Learning Repository Irvine, CA," *University of California*, http://www.ics.uci.edu/ mlearn/MLRepository.html, (2007).