Validation of Region-based Crossover for Clustering Problems

Jeevan F. D'Souza, C. Kelly Adams and Andrew Reed

Department of Engineering and Information Sciences
DeVry College of New York,
180 Madison Ave, Suite 900,
New York NY 10016
Email: jdsouza@devry.edu, ckadams@devry.edu, reed.andrew@gmail.com

ABSTRACT

The k-means algorithm is a widely used partitional clustering algorithm because of its simplicity and computational efficiency. One problem with the k-means algorithm is that the quality of partitions produced is highly dependent on the initial selection of centers. The problem of center selection has been tackled in the past using genetic algorithms (GA). One of the most effective GA for k-means clustering is the region-based genetic algorithm (RBGA). This research aimed at assessing the RBGA across a variety of cluster representatives and distance metrics. The experimental results show the superior performance of the RBGA, as compared to other popular genetic algorithm approaches, indicating that region-based crossover may prove an effective strategy across a broad range of clustering problems.

Keywords: k-means algorithm, clustering, genetic algorithm, crossover operation, center selection

Mathematics Subject Classification: 62H30

1. INTRODUCTION

Clustering is the process of grouping data into clusters based on distance between or similarities of the data objects. Clustering has applications in biology, medicine, market research, social network analysis, software evolution, image segmentation, data mining, search result grouping, and other areas (Jain & Dubes, 1988).

The Sum of the Squared Error (SSE) is a commonly used metric to calculate the quality of the partitions created in clustering algorithms (Laszlo & Mukherjee, 2007). SSE is the sum of the squared distance between each point and the center of the cluster to which the point is assigned. Consider \( x_i \) to be the \( i \)th data point in a cluster of \( n \) points and \( \bar{x} \) to be the centroid of its cluster. Then the SSE of the cluster would be defined as:

\[
\sum_{i=1}^{n} (x_i - \bar{x})^2
\]
Now consider there to be a total of $k$ clusters in the search space. Then the total SSE ($SSE_T$) of the partition would be defined as:

$$
SSE_T = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2
$$

where $x_{ij}$ represents the $i^{th}$ data point in the $j^{th}$ cluster and $\bar{x}_j$ is the centroid of the $j^{th}$ cluster. Clustering seeks to minimize $SSE_T$. A high-quality cluster will have low $SSE_T$.

The $k$-means algorithm is a commonly used partitional clustering algorithm (Wagstaff, Cardie, Rogers, & Schrödl, 2001). The $k$-means algorithm is as follows: (Al-shboul & Myaeng, 2009).

**Input:** A set of data points  
**Output:** A partition with $k$ clusters  

$k$ distinct data points are chosen as cluster centers.  

Repeat the following steps until the centroids do not change (steady state is reached)  
- Every data point is assigned to only a single closest center based on its distance from the center using a distance metric  
- Once these $k$ temporary clusters are determined, the centroids of these clusters are calculated using a distance metric  
- The centers are repositioned to the location of the centroids  

The main problem with the $k$-means algorithm is that the quality of its solutions is sensitive to the initial selection of cluster centers (Zhang, & Xia, 2009). The problem of center selection has been tackled using genetic algorithms (GA) since they are useful in generating solutions to search problems.

GAs use concepts of evolutionary biology like crossover, mutation, selection and inheritance to generate “fit” offspring. In GAs a population of strings called chromosomes is initially selected. These chromosomes encode candidate solutions (individuals) to the problem. In most cases individuals are represented as binary strings, but other representations are also used. The search spans successive generations. In each generation, the fitness of the individuals is measured. Genetic operations like crossover and mutation are carried out on these individuals to generate new offspring. The selection process ensures that the fittest offspring are most likely to be used as parents in the next generation. The process is continued until a termination condition is reached. Since genetic algorithms are constantly looking for fittest solutions, they often rapidly generate solutions to search problems as compared to other methods.

A simple generational genetic algorithm is shown below:

**Select an initial population of individuals**  
**Evaluate the fitness of each individual in that population**  
**Repeat the following steps on this generation until termination condition is reached**  
- Select individuals for reproduction based on fitness  
- Breed new individuals (offspring) through crossover and mutation  
- Evaluate the fitness of new individuals  
- Replace least-fit individuals in the population with new fittest individuals
In this research, an individual represents a collection of points in space. Specifically, each individual represents a set of \( k \) centers, and each center is represented by a sequence of \( n \) floating-point values where the space is \( n \)-dimensional. An individual represents an initial configuration for running \( k \)-means.

(Babu & M, 1993; Bandyopadhyay & Maulik, 2002; Bhuyan, Raghavan, & Venkatesh, 1991; Lu, Lu, Fotouhi, Deng, & Brown, 2004; Krishna & Murthy, 1999; Laszlo & Mukherjee, 2006; Maulik & Bandyopadhyay, 2000; Laszlo & Mukherjee, 2007) have used GAs to tackle the initial center selection problem of \( k \)-means. Of these approaches, the Region-Based Genetic Algorithm (RBGA) approach used by (Laszlo & Mukherjee, 2007) has generated partitions of best quality. The RBGA used the Euclidean distance as the distance metric and centroid as the representative object of the cluster (ROC). The distance metric and ROC jointly determine the objective function.

The goal of this research was to assess the RBGA approach using different objective functions. RBGA was compared with other popular GAs proposed by (Babu & Murthy, 1993) and (Krishna & Murthy, 1999). The results obtained showed the superior performance of the RBGA across all objective functions, indicating that it may be an effective strategy across a broad range of clustering problems.

The rest of this paper is organized as follows: Section 2 gives a brief description of the GAs used in this study. Section 3 describes the metrics and ROCs used in this study. Section 4 discusses the results. Section 5 presents conclusions, implications and future research recommendations.

2. METRICS AND CLUSTER REPRESENTATIVES USED IN THIS STUDY

2.1 Metrics

A distance metric is a function that defines the distance between two elements of a set in space (Deza & Deza, 2009). A metric could be used to determine a distance between two points or objects. Depending on the type of problem, certain metrics prove useful. The Euclidean metric is most commonly used to calculate distances in clustering problems (Deza & Deza, 2009). But other metrics are also useful. The Euclidean, Manhattan, Chebyshev and Minkowski distances are the most commonly used distance metrics for clustering and were selected as distance metrics for the experiments in this research (Deza & Deza, 2009).

The Euclidean distance or Euclidean metric is the distance between two points that is represented by a straight line in Cartesian space which is also the shortest distance between two points. The Euclidean distance is also called “ordinary” distance in some cases. The Euclidean distance can be calculated by the Pythagorean formula or the distance formula (Deza & Deza, 2009). The Euclidean distance formula can be used to determine the distance between two points with any number of dimensions. If \( x = (x_1, x_2, \ldots, x_n) \) and \( y = (y_1, y_2, \ldots, y_n) \) are two \( n \)-dimensional points in Euclidean \( n \)-space, then the distance from \( x \) to \( y \) is given by:

\[
d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
\]
The Manhattan distance or taxicab distance is the distance of two points measured in a grid-based co-ordinate system (Deza & Deza, 2009). The distance gets its name from the grid-like city streets of Manhattan that taxicabs usually have to drive through to get from point $A$ to point $B$. The Manhattan distance is not the shortest path between two points. In an $n$-dimensional plane, the Manhattan distance between $x$ and $y$ is:

$$
\sum_{i=1}^{n} |x_i - y_i|
$$

The Chebyshev distance (Deza & Deza, 2009) is a metric where the distance between two points is the greatest of their differences along any co-ordinate dimension. In order to use this metric, the formula to calculate distance will be as follows: In an $n$-dimensional plane, the Chebyshev distance between $x$ and $y$ is:

$$
D_{\text{cheb}}(x,y) = \max_i \{ |x_i - y_i| \}
$$

A distance metric that generalizes all of the above-mentioned distance metrics is called the Minkowski distance metric (Deza & Deza, 2009). For two $n$-dimensional points $x$ and $y$ the Minkowski distance would be given by:

$$
D_p(x,y) = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}
$$

By varying the value of $p$, the Minkowski distance metric yields the other metrics. If $p=1$, the Minkowski metric is the same as the Manhattan distance. If $p=2$, the Minkowski metric is the same as the Euclidean distance. If $p=\infty$, the Minkowski metric is the same as the Chebyshev distance.

**2.2 Cluster Representatives**

A centroid or geometric center is the arithmetic mean of all points of the object or shape (Laurini & Thompson, 1992; Kekre & Sarode, 2008). A centroid can be extended to any shape or object in an $n$-dimensional space. In an $n$-dimensional space, it could be defined as the intersection of every hyper-plane that divides the object or shape into two parts of equal moment. Consider $kn$-dimensional points where $d_i$ denotes the $i^{th}$ point in the dataset. The centroid $C$ of that dataset would be given as:

$$
C = \frac{1}{k} \sum_{i=1}^{k} d_i
$$

Besides centroids, medoids are also a common representative for clusters. A medoid is a point of the cluster whose average dissimilarity to all the objects in the cluster is minimal (Zhang, Ramakrishnan, & Livny, 1998). Note that a medoid, unlike a centroid, is always a data point of the cluster. The true medoid is the data point that minimizes the cluster’s SSE (i.e., average dissimilarity is minimal). The data point closest to the centroid is often used as a heuristic for the medoid to avoid time complexity.

A common application of the medoid is the $k$-medoids clustering algorithm. The $k$-medoids clustering algorithm is similar to the $k$-means algorithm and is a partitional clustering algorithm. The most common $k$-medoids algorithm used is the Partitioning Around Medoids (PAM) algorithm (Kaufman & Rousseeuw, 2008). The goal is to find a subset $\{m_1, \ldots, m_k\} \subset \{1, \ldots, n\}$ which minimizes the objective function.
Each object is then assigned to the cluster corresponding to the nearest medoid. That is, object $i$ is put into cluster $v_t$ when medoid $m_{v_t}$ is nearer to $i$ than any other medoid $m_w$, $d(i, m_{v_t}) < d(i, m_w)$ for all $w = 1, \ldots, k$.

3. EXPERIMENTS AND RESULTS

3.1 Experiments performed

The algorithms presented in this research were tested, verified and analyzed using a C++ program running on a Personal Computer (PC). The program inputted data sets from text files stored on the hard drive and ran the chosen methods on these data by varying the number of centers, ROCs, and the choice of metric. A list of methods, datasets, metrics, and centers that were chosen while running the program are listed below:

1) Method
   a. $k$-means algorithm (centroids only) (Al-shboul & Myaeng, 2009)
   b. $k$-medoids algorithm (medoids only) (Kaufman & Rousseeuw, 2008)
   c. GA without crossover using $k$-means (Krishna & Murthy, 1999)
   d. GA with random crossover using $k$-means (Babu & Murthy, 1993)
   e. GA with region-based crossover using $k$-means (Laszlo & Mukherjee, 2007)
   f. GA without crossover using $k$-medoids (Krishna & Murthy, 1999)
   g. GA with random crossover using $k$-medoids (Babu & Murthy, 1993)
   h. GA with region-based crossover using $k$-medoids (Laszlo & Mukherjee, 2007)

2) Datasets:
   a. German town data (GTD) (Spath, 1980)
   b. British town data (BTD) (Spath, 1980)
   c. TSP-LIB-1060 (Reinelt, 1991)
   d. TSP-LIB-3038 (Reinelt, 1991)

3) Distance measure
   a. Euclidean metric (Minkowski metric with $p=2$)
   b. Manhattan metric (Minkowski metric with $p=1$)
   c. Chebyshev metric (Minkowski metric with $p=\infty$)
   d. Minkowski metric with value of $p=3$

4) Number of centers $k$

Based on selected parameters, the program computed the best and average SSE$_T$ and saved the information in a Microsoft Excel file. The program was then reset and run with a different set of parameters.

Four benchmark data sets were chosen and are outlined in Table 1. The best known partitions using these datasets have been published by (Babu & Murthy, 1993; Krishna & Murthy, 1999; Laszlo & Mukherjee, 2007)
Table 1. Datasets used in this Research

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Number of Points</th>
<th>Number of Dimensions</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>German town data (GTD)</td>
<td>59</td>
<td>2</td>
<td>Small</td>
</tr>
<tr>
<td>British town data (BTD)</td>
<td>50</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>TSP-LIB-1060</td>
<td>1060</td>
<td>2</td>
<td>Large</td>
</tr>
<tr>
<td>TSP-LIB-3038</td>
<td>3038</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Since GAs are stochastic, they produced different partitions with the same parameter set. For this reason, multiple runs were performed on each parameter set and the average and best value of SSE per generation was computed and stored. $k$-means and $k$-medoids were run with varied number of centers, the same total number of times as they were used in the GA. For example, if we performed $R$ runs of each GA for $N$ generations with population size $P$, then $k$-means and $k$-medoids was run $RNP$ many times, each with a varied initial set of centers.

The main criterion used to compare the region-based crossover GA to other methods was the optimization of the objective function, which is the function used to compute fitness values. The goal was to have this sum of the partitions (total SSE) as small as possible. Experiments were conducted on the four benchmark datasets mentioned in Table 1. The BTD and GTD datasets are smaller in size and are referred to as the small datasets. TSP-LIB-1060 and TSP-LIB-3038 are the large datasets. Table 2 shows the number of centers ($k$) chosen for the datasets.

Table 2. Choice of Centers for each Dataset used in this Research

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of Centers ($k$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTD</td>
<td>4 – 10</td>
</tr>
<tr>
<td>BTD</td>
<td>4 – 10</td>
</tr>
<tr>
<td>TSP-LIB-1060</td>
<td>10, 100, 150</td>
</tr>
<tr>
<td>TSP-LIB-3038</td>
<td>10, 100, 500</td>
</tr>
</tbody>
</table>

The values chosen for the number of centers was based on the total number of data points in the datasets and also on experiments carried out in (Laszlo & Mukherjee, 2007). For each set of centers, experiments were carried out using the eight different methods and the four different distance metrics mentioned above.

For the small datasets, the six GAs were run 10 times, for 100 generations with a population size of 100, with mutation probability of 0.001 and crossover probability of 0.8, for each parameter set. In addition, for comparison, $k$-means and $k$-medoids were separately run 100,000 times for each parameter set for analysis and comparison. This number is equal to the number of times $k$-means ($k$-medoids) is run by the GA; it is equal to the product of the number of generations, the population size and total number of runs used in the GAs.

For the large datasets, the six GAs were run 4 times, for 100 generations with a population size of 20, with mutation probability of 0.001 and crossover probability of 0.8 for $k=10$, for each parameter set. In addition, for comparison, $k$-means and $k$-medoids were separately run 8,000 times for each parameter set. The six GAs were run 4 times, for 20 generations with a population size of 20, with mutation
probability of 0.001 and crossover probability of 0.8 for \( k > 10 \), for each parameter set. \( k \)-means and \( k \)-medoids were run separately 1,600 times for each parameter set.

The \( k \)-means algorithm was used when centroids were used as the ROC. When medoids were used as the ROC, the \( k \)-medoid algorithm PAM was used. The medoid was chosen by finding the centroid and then locating the closest data point to it, to reduce complexity. The naming convention used to describe the six GA methods is shown in Table 3.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Name used</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA without crossover using ( k )-means</td>
<td>GA1-means</td>
</tr>
<tr>
<td>GA with random crossover using ( k )-means</td>
<td>GA0-means</td>
</tr>
<tr>
<td>GA with region-based crossover using ( k )-means</td>
<td>GA2-means</td>
</tr>
<tr>
<td>GA without crossover using ( k )-medoids</td>
<td>GA1-medoids</td>
</tr>
<tr>
<td>GA with random crossover using ( k )-medoids</td>
<td>GA0-medoids</td>
</tr>
<tr>
<td>GA with region-based crossover using ( k )-medoids</td>
<td>GA2-medoids</td>
</tr>
</tbody>
</table>

### 3.2 Data collected

#### 3.2.1 Small datasets

The data collected for the small data sets for each parameter set of all methods included:

1) Best \( \text{SSE}_T \) after all runs
2) The average (over all runs) of the average \( \text{SSE}_T \) per generation for the GAs

Bar charts were created to compare best \( \text{SSE}_T \) after all runs for the GAs. Line plots were created to analyze data collected for the average \( \text{SSE}_T \) per generation for the GAs. Figures 1 and 2 show samples of these plots and charts.

![Figure 1](image_url)
Figure 2. Average $SSE_T$ per Generation using the GTD Dataset, $k$-means, and the Chebyshev Metric for $k=9$

For GTD data set, there was no difference in best $SSE_T$ values calculated by all methods for a given parameter set. GA2 improved more rapidly on average, per generation, than the other two methods, and converged to a lower SSE value. The same results were observed for GA1 over GA0. $k$-means and $k$-medoids typically took many more runs to produce solutions as good as, or close to, the GAs for all parameter sets, as compared to the number of $k$-means/$k$-medoids runs required to reach optimum solutions by the GAs.

For the BTD data set, there was no difference in best $SSE_T$ values calculated by all methods for a given parameter set for $k=4$ to $k=8$. GA2 found better solutions for $k=9$ and $k=10$ in most cases. Again it was clearly observed that GA2 always found the best $SSE_T$ values in fewer generations than the other methods. It was also observed that GA1 found these solutions in fewer generations than GA0. Once again, $k$-means and $k$-medoids typically took many more runs to produce solutions as good as, or close to, the GAs for all parameter sets, as compared to the number of $k$-means/$k$-medoids runs required to reach optimum solutions by the GAs.

In the line plots, there was not much difference in the values of the three GAs for $k=4$ to $k=8$. For $k=9$ and $k=10$ most plots looked like the one shown in Figure 2. In most cases, the three GAs start out with average $SSE_T$ values that are close to each other in the first generation. A sharp fall in average $SSE_T$ values for the first 15 generations for all GAs occurs. During this phase GA1 typically outperformed GA0 and GA2. From generations 15-100 GA1 showed less progress than the other two GAs. GA2 always ended up showing best average $SSE_T$ values for these generations.

The small data sets have relatively few data points and hence are easy to cluster especially when the numbers of centers are low. Therefore it was possible to get best known $SSE_T$ values in most cases within a few runs of $k$-means/$k$-medoids or a few generations of the GAs. As the number of centers approached 10, the clustering seemed to get more challenging and the superiority of GA2 could be seen. Since the BTD dataset has four dimensions as compared to the two-dimensional GTD dataset, it was harder to arrive at good solutions for $k>=9$. For example, $k$-means took only two runs to arrive at
the best $SSE_T$ value for the GTD dataset using the Euclidean metric for $k=4$ and over 80,000 runs to do the same for the BTD dataset for $k=10$.

### 3.2.2 Large Datasets

The data collected for the large data sets for each parameter set of all methods included:

1) Best $SSE_T$ after all runs
2) The average (over all runs) of the average $SSE_T$ per generation for the GAs

Bar charts were created to compare best $SSE_T$ after all runs for the GAs. Line plots were created to analyze data collected for the average $SSE_T$ per generation for the GAs. Examples of these charts and plots are shown below in Figures 3, 4 and 5.

**Figure 3.** Best $SSE_T$ values using the TSP-LIB-3038 Dataset, $k$-medoids, and the Minkowski Metric for $k=500$
For the larger datasets, GA2 almost always found better solutions than the other methods. GA0 found better solutions than GA1 for most parameter sets. k-means and k-medoids typically did not find solutions as good as the GAs for all parameter sets. For the large datasets similar results as Figure 4 were seen for $k=10$. For values of $k>10$, the superiority of GA2 was clearly evident across all metrics and ROCs.

Since the large data sets have many data points, they are difficult to cluster especially when the number of centers is high. GA2 almost always outperformed the other GAs when large datasets were used. Region-based crossover, replacing unused empty centers with random data points and using $k$-means as a genetic operator were instrumental in GA2’s superior performance. GA0, in most cases, gave second best solutions because GA0 performed bit-wise crossover and mutation after converting the centers to binary bit-strings. The down-side of GA0 was the running time. GA1 had the fastest running time but the poorest quality of partitions mainly because the emphasis was on time rather than quality of solutions. (Krishna & Murthy, 1999) show that GA1 generates good quality partitions using the small datasets. GA1 has not been run on the large datasets before. In most cases, this research proved that the quality of solutions produced by GA1 was poor on large datasets as compared to GA0 and GA2. GA1 gave better results than $k$-means/k-medoids.

### 3.3 Findings

Tables 4 and 5 show the ratios of the average of the average (over all runs) of the average SSE$_T$ per generation for the GAs for the large datasets.
Table 4. Ratio of Averages of GA0/GA2 and GA1/GA2 for Varied Metrics, Centers and ROCs for the TSP-LIB-1060 Dataset

<table>
<thead>
<tr>
<th>k</th>
<th>Metric</th>
<th>GA0/GA2 Means</th>
<th>GA1/GA2 Means</th>
<th>GA0/GA2 Medoids</th>
<th>GA1/GA2 Medoids</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Euclidean</td>
<td>1.000378143</td>
<td>1.000495423</td>
<td>1.005981089</td>
<td>1.014930742</td>
</tr>
<tr>
<td></td>
<td>Manhattan</td>
<td>1.001115032</td>
<td>1.000757373</td>
<td>1.007109589</td>
<td>1.013369941</td>
</tr>
<tr>
<td></td>
<td>Chebyshev</td>
<td>1.000170349</td>
<td>1.001114881</td>
<td>1.012449041</td>
<td>1.026830410</td>
</tr>
<tr>
<td></td>
<td>Minkowski</td>
<td>1.000260470</td>
<td>1.000100256</td>
<td>1.008218273</td>
<td>1.009654326</td>
</tr>
<tr>
<td>100</td>
<td>Euclidean</td>
<td>1.035089376</td>
<td>1.036455117</td>
<td>1.015616627</td>
<td>1.020009708</td>
</tr>
<tr>
<td></td>
<td>Manhattan</td>
<td>1.007511521</td>
<td>1.030327665</td>
<td>1.009315944</td>
<td>1.018741972</td>
</tr>
<tr>
<td></td>
<td>Chebyshev</td>
<td>1.012352897</td>
<td>1.003297601</td>
<td>1.030117768</td>
<td>1.046529561</td>
</tr>
<tr>
<td></td>
<td>Minkowski</td>
<td>1.016807201</td>
<td>1.025093157</td>
<td>1.029096752</td>
<td>1.053722271</td>
</tr>
<tr>
<td>150</td>
<td>Euclidean</td>
<td>1.013376847</td>
<td>1.010324399</td>
<td>1.019688274</td>
<td>1.019013600</td>
</tr>
<tr>
<td></td>
<td>Manhattan</td>
<td>1.009285565</td>
<td>1.026201993</td>
<td>1.025139773</td>
<td>1.034304341</td>
</tr>
<tr>
<td></td>
<td>Chebyshev</td>
<td>1.029742502</td>
<td>1.023408621</td>
<td>1.037398957</td>
<td>1.040298133</td>
</tr>
<tr>
<td></td>
<td>Minkowski</td>
<td>1.041993845</td>
<td>1.034413701</td>
<td>1.013752644</td>
<td>1.003080061</td>
</tr>
</tbody>
</table>

Table 5. Ratio of Averages of GA0/GA2 and GA1/GA2 for Varied Metrics, Centers and ROCs for the TSP-LIB-3038 Dataset

<table>
<thead>
<tr>
<th>k</th>
<th>Metric</th>
<th>GA0/GA2 Means</th>
<th>GA1/GA2 Means</th>
<th>GA0/GA2 Medoids</th>
<th>GA1/GA2 Medoids</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Euclidean</td>
<td>1.000061633</td>
<td>1.000127004</td>
<td>1.001282062</td>
<td>1.006047463</td>
</tr>
<tr>
<td></td>
<td>Manhattan</td>
<td>1.000532325</td>
<td>1.000887831</td>
<td>1.005018959</td>
<td>1.006082527</td>
</tr>
<tr>
<td></td>
<td>Chebyshev</td>
<td>1.000020108</td>
<td>1.000656688</td>
<td>1.004929617</td>
<td>1.007982057</td>
</tr>
<tr>
<td></td>
<td>Minkowski</td>
<td>1.000078451</td>
<td>1.000113116</td>
<td>1.002887259</td>
<td>1.001432746</td>
</tr>
<tr>
<td>100</td>
<td>Euclidean</td>
<td>1.002705900</td>
<td>1.00681437</td>
<td>1.012009741</td>
<td>1.005883685</td>
</tr>
<tr>
<td></td>
<td>Manhattan</td>
<td>1.007366658</td>
<td>1.006942475</td>
<td>1.015557108</td>
<td>1.014583853</td>
</tr>
<tr>
<td></td>
<td>Chebyshev</td>
<td>1.007101395</td>
<td>1.018161046</td>
<td>1.013556616</td>
<td>1.013607822</td>
</tr>
<tr>
<td></td>
<td>Minkowski</td>
<td>1.006569237</td>
<td>1.009022603</td>
<td>1.018520211</td>
<td>1.014954022</td>
</tr>
<tr>
<td>500</td>
<td>Euclidean</td>
<td>1.020746628</td>
<td>1.012518283</td>
<td>1.051801952</td>
<td>1.045031418</td>
</tr>
<tr>
<td></td>
<td>Manhattan</td>
<td>1.007167205</td>
<td>1.01344834</td>
<td>1.029465240</td>
<td>1.021863563</td>
</tr>
<tr>
<td></td>
<td>Chebyshev</td>
<td>1.012223678</td>
<td>1.015234882</td>
<td>1.005617342</td>
<td>1.013492604</td>
</tr>
<tr>
<td></td>
<td>Minkowski</td>
<td>1.005841592</td>
<td>1.015141917</td>
<td>1.007651413</td>
<td>1.012808735</td>
</tr>
</tbody>
</table>

From Tables 4 and 5 it is evident that GA2 performed better than all the other methods across all sets of parameters. Best SSE data and average SSE per generation data show that GA2 performed better than all the other methods across all sets of parameters. The superior results of GA2 were more pronounced in the larger datasets due to the challenge of searching significantly larger search spaces. The main reasons for GA2’s superior performance were region-based crossover, replacement of empty centers and using k-means as a genetic operator. These three strategies are present in GA2, but absent from GA0 and GA1. In GA2, the crossover operator exchanged centers...
that occupy a specific region of space. In spatial data clustering, center selection is spatially decomposable allowing centers in a specific region of space to make better candidates for crossover rather than random centers which leads to significantly better solutions as compared to other crossover methods that exchange random centers. Moving chromosome centers/genes to new positions after running $k$-means and replacing newly positioned centers that do not serve as centroids, helps keep the search focused on finding good partitions and reducing the possibility of having useless centers. Also, the superior performance of GA2 became more evident as the number of centers was increased across all datasets because region-based crossover yielded sets of centers that converge with less iteration under $k$-means/$k$-medoids.

GA1 ran faster than other methods across all sets of parameters mainly because of using one-step of $k$-means/$k$-medoids instead of crossover. GA1 did not require re-calculating the fitness by re-running $k$-means/$k$-medoids every time only crossover was performed. Fitness needed to be recalculated only when mutation occurred which was very rare. Fitness calculation dominated running time of the GAs.

GA1 almost always performed better than the other GA methods in the first few generations but then showed little or no evolution after 10 generations because using one step of $k$-means for crossover is a good approach for speeding up the GA but not for providing promising solutions over several generations.

GA0 showed better performance than GA1 across most sets of parameters but took more running time than all other methods because centers are reshuffled with every random crossover, so $k$-means/$k$-medians took many more iterations to converge.

There was no dependence between GA0 and choice of metric with respect to quality of solution. The same was true for GA1 and GA2 and can be seen from the data in Tables 4 and 5 by comparing the “average SSE$_T$ per generation” line plots across different metrics.

There was a slight dependence between the GAs and ROCs in the larger datasets for $k=10$. The superiority of GA0 and GA2 was a little more pronounced when medoids were used instead of means and was observed in the “average SSE$_T$ per generation” line plots across different ROCs. For other values of $k$ there was no dependence between a GA and a ROC and was observed from the data in Tables 4 and 5 and once again by comparing the “average SSE$_T$ per generation” line plots across different ROCs.

The best SSE$_T$ values presented in this research did not always match the best-known SSE$_T$ values in (Laszlo & Mukherjee, 2007), because the GAs were not run for as many generations with large population sizes. Determining the best SSE$_T$ values for a given parameter set was not the goal of this research. The goal of this research was to prove that GA2 performs better than other methods across a range of datasets, metric, ROCs and number of centers. The goal has been accomplished and is evident in the results.

4. DISCUSSION AND CONCLUSIONS

This research aimed at proving that the region-based genetic algorithm (RBGA) is a promising algorithm for partitional $k$-means clustering across a range of objective functions. (Laszlo &
Mukherjee, 2007) proved that the region-based method performed better than other GA methods when centroids were used as the representative object of the cluster and the Euclidean metric was used as distance metric. This research went further in the assessment of the region-based approach by varying the range of objective functions, and successfully proved that it performs better than other methods across all sets of parameters on four benchmark datasets.

Across all these parameters the RBGA found better or comparable results than the other methods. The superior performance of the RBGA was more pronounced in the large datasets that consisted of larger search spaces and more data points. Superior performance was also seen when the number of centers ($k$) was higher for all datasets.

The criterion used to measure the performance of all methods was the total SSE of all the partitions of the search space. The lower the value of $\text{SSE}_T$, the better is the performance. The data acquired in this research included best $\text{SSE}_T$ values for all methods and average (over all runs) of the average $\text{SSE}_T$ values per generation for the GAs. Data was plotted into bar graphs and line plots. From the 80 line plots and 48 bar graphs obtained, it is clearly evident that the RBGA found better partitions than other methods across all sets of parameters for all four datasets that were used in the study.

This study implies that using genetic algorithms to solve the center selection problem of $k$-means is promising over a range of metric and ROCs because of the selection process of a GA which selects the fittest individuals to breed in the next generation, as opposed to $k$-means / $k$-medoids which randomly select centers for every run. This research also reinforces the concept stated by (Laszlo & Mukherjee, 2007), that the center selection problem of $k$-means is spatially decomposable to a large extent. The RBGA makes use of spatial decomposition of the search space by creating regions and performing crossover between chromosome centers in a specific region which led to significant improvement in the partitions produced.

Spatial decomposition could be applied to other steps of a genetic algorithm like initialization, mutation, selection and fitness calculation to produce better results. Spatial decomposition could also be directly applied to the $k$-means algorithm so that the initial set of centers could be spatially decomposed into regions rather than randomly selected. This paper used four datasets that had data points up to four dimensions. It would be interesting to see the nature of the GA methods as higher dimensions are used. This study was limited to four distance metric and two ROCs. Other distance metrics and ROCs could be used to assess the GAs. Some recommendations for metrics are the Pearson correlation metric, and the Spearman correlation metric. Some recommendations for ROCs are medians, the Chebyshev center, and geometric mean. This research proves that spatially decomposing the search space into half-spaces significantly improves GA performance. Such half-spaces might not be the ideal region within which crossover is performed. Different methods of spatial decomposition should be tried and tested.

5. REFERENCES


