Genetic Algorithms with the Crossover-Like Mutation Operator for the k-Means Problem

Lev Kazakovtsev[1,2](✉), Guzel Shkaberina[1], Ivan Rozhnov[1,2], Rui Li[1], and Vladimir Kazakovtsev[3]

[1] Reshetnev Siberian State University of Science and Technology, prosp. Krasnoyarskiy Rabochiy 31, Krasnoyarsk 660031, Russia
levk@bk.ru

[2] Siberian Federal University, prosp. Svobodny 79, Krasnoyarsk 660041, Russia

[3] ITMO University, Kronverksky pr. 49, St. Petersburg 197101, Russia

Abstract. Progress in the development of automatic grouping (clustering) methods, based on solving the p-median and similar problems, is mainly aimed at increasing the computational efficiency of the algorithms, their applicability to larger problems, accuracy, and stability of their results. The researchers' efforts are focused on the development of compromise heuristic algorithms that provide a fairly quick solution with minimal error. The Genetic Algorithms (GAs) with greedy agglomerative crossover procedure and other special GAs for the considered problems demonstrate the best values of the objective function (sum of squared distances) for many practically important problems. Usually, such algorithms do not use any mutation operator, which is common for other GAs.

We propose new GAs for the k-means problem, which use the same procedures as both the crossover and mutation operators. We compared a simple GA for the k-means problem with one-point crossover and its modifications with the uniform random mutation and our new crossover-like mutation. In addition, we compared the GAs with greedy heuristic crossover procedures to their modifications which include the crossover-like mutation. The comparison results show that the idea of our new mutation operator is able to improve significantly the results of the simplest GA as well as the genetic algorithms with greedy agglomerative crossover operator.

Keywords: Clustering · k-Means · Genetic algorithm · Greedy agglomerative procedure

1 Introduction and Problem Statement

The k-means problem [1] can be described as finding a set of k cluster centroids $X_1, ...X_k$ in a d-dimensional space with the minimal sum of squared distances

Supported by the Ministry of Science and Higher Education of the Russian Federation (Project FEFE-2020-0013).

from them to the given N points (vectors) A_i (SSE, sum of squared errors):

$$\arg \min_{X_1,...,X_k \in \mathbb{R}^d} F(X_1, ..., X_k) = \sum_{i=1}^{N} \min_{j \in \{\overline{1,k}\}} \|X_j - A_i\|^2 . \qquad (1)$$

In the continuous p-median problem, a sum of distances (instead of squared distances) is calculated, and the searched points are called centers or medians. If a sum of Manhattan (L_1, rectilinear) distances is used as the minimized function, the problem is referred to as the k-means problem [2].

An algorithm of the same name (Algorithm 1) [3,4], also known known as Lloyd algorithm, sequentially improves a known solution, looking for a local minimum of (1). This local search algorithm (LSA) is simple, fast, and applicable to the widest class of problems. The algorithm has a limitation: the number of groups (clusters) k must be known. The result is highly dependent on the initial solution chosen at random.

Algorithm 1. k-Means (Lloyd, ALA: Alternating Location-Allocation)

```
Require: data vectors $A_1...A_N$, $k$ initial cluster centers (centroids) $X_1, ..., X_k$.
  repeat
    Step 1: For each of centers $X_i$, compose clusters $C_i$ of data vectors so that each
    of the data vectors is assigned to the nearest center.
    Step 2: Calculate new center $X_i$ for each of the clusters.
  until Steps 1,2 result in no modifications.
```

Both Steps 1 and 2 improve the objective function (1) value.

In this research, we try to improve the accuracy of the k-means problem result (1) and its stability within a fixed, limited run time. By the accuracy of the algorithm, we mean the achieved value of (1). We do not consider other important issues in the fields of cluster analysis such as adequacy of the model (1) and correspondence of the algorithm result to the actual partition [5].

The idea of Genetic Algorithms (GAs) is based on a recombination of elements of some candidate solutions set called "population". Each candidate solution is called an "individual" encoded by a "chromosome" represented by a vector of bits, integers or real numbers depending on the algorithm. In the modern literature, there is practically no systematization of the approaches used (see [6–8]), for algorithms with the real-number (centroid-based) chromosome encoding.

The first GA for the discrete p-median problem was proposed by Hosage and Goodchild [9]. Algorithm presented in [10] gave more precise results with a very slow convergence. In [11], the authors proposed a faster algorithm with a special "greedy" heuristic crossover operator which is also precise. All these algorithms solve discrete problems (p-median problem on a network) and use a simple binary chromosome encoding (1 for the network nodes selected as the medians and 0 for those not selected).

The mutation operator is devoted to guarantee the GA population diversity [12]. Usually, for the k-means and similar problems, the mutation randomly changes one or many chromosomes, replacing some centroids [12–14] or assignment of an object. For example, in [13] authors proposed the distance-based mutation which changes an allele value (an allele is a part of a chromosome that encodes the assignment an object to a cluster) depending on the distances of the cluster centroids from the corresponding data point. Each allele corresponds to a data point and its value represents the corresponding cluster number. The mutation operator is defined such that the probability of changing an allele value to a cluster number is higher if the corresponding centroid is closer to the data vector. To apply the mutation operator to the allele $s_W(i)$ corresponding to centroid X_i, let us denote $d_j = \|X_i - A_j\|$ where A_j is a data vector. Then, the allele is replaced with a value chosen randomly from the following distribution: $p_j = Pr\{s_W(i) = j\} = (c_m d_{max} - d_j)/(\sum_{i=1}^{K}(c_m d_{max} - d_i))$ where c_m is a constant usually $\leq$1 and $d_{max} = \max\{d_j\}$. In the case of a partition with one or more than one singleton clusters, the above mutation may result in the formation of empty clusters with a non-zero probability. It may be noted that the smaller the number of clusters, the larger is the SSE measure; so empty clusters must be avoided [13].

If the cluster centroids are searched in a continuous space, some GAs still use the binary encoding [15–17]. In Algorithm 1, the initial solutions are usually subsets of the data vectors set. Thus, in the chromosome code, 1 means that the corresponding data vector must be used as the initial centroid, and 0 for those not selected. In this case, some LSA (Algorithm 1 or similar) is used at each iteration of the GA. In the GAs for the k-means and analogous problems, which use the traditional binary chromosome encoding, many mutation techniques can be used. For example, in [18], the authors use binarization and represent the chromosome with binary strings composed of binary-encoded features (coordinates) of the centroids. The mutation operator arbitrarily alters one or more components (binary substrings) of a selected chromosome. In [19,20], authors call their algorithms "Evolutionary k-Means." However, they actually solve an alternative problem related to the k-Means problem aimed to increase clustering stability. This algorithm operates with the binary consensus matrices and uses two types of the mutation operators: cluster split (dissociative) and cluster merge (agglomerative) mutation. In [21], the chromosomes are strings of integers representing the cluster number for each of clustered objects, and the authors solve the k-means problem with simultaneous determining the number of clusters based on the silhouette [22] and David-Bouldin criteria [23] (similar approach is used in a much simpler algorithm X-Means [24]) which are used as the fitness functions. Thus, in [21], authors solve a problem with the mathematical statement other than (1) and use cluster recalculating in accordance with (1) as the mutation operator. Similar encoding is used in [13] where authors propose a mutation operator, which changes the assignment of individual data objects to clusters.

In [14], the authors encode the solutions (chromosomes) in their GA as sets of centroids represented by their coordinates (vectors of real numbers) in a d-dimensional space. The same principle of centroid-based chromosome representation is used in the GAs of the Greedy Heuristic Method [25]. In [14], the mutation procedure is as follows. Randomly generate a number from 0 to 1. If the number is less than mutation probability μ, the chromosome will mutate. The number $b \in (0, 1]$ is randomly generated with the uniform distribution. If the position of a centroid is v, the mutation is as follows:

$$v \leftarrow \begin{cases} v \pm 2 \times b \times v, & v \neq 0, \\ v = v \pm 2 \times b, & v = 0. \end{cases} \tag{2}$$

Signs "+" and "−" have the same probability here [18].

In (2), coordinates of a centroid are shifted randomly. A similar shifting technique with an "amplification factor" was used in [26,27]. However, the local minima distribution among the search space is not uniform [12]: new local minima of (1) can be found with higher probability in some neighborhood of a given local minimum than in a neighborhood of a randomly chosen point (here, we do not mean an ϵ-neighborhood). Thus, mixing the centroids from two local minima must usually outperform the random shift of centroid coordinates. The idea of combining local minima is the basic idea of the GAs with the greedy agglomerative heuristic crossover procedures [25] and other algorithms [28] which use no mutation operator. Such algorithms are able to demonstrate more accurate results in comparison with many other algorithms for many practical problems. However, one of the most important problems of the GAs is the convergence of the entire population into some narrow area (population degeneration) around some local minimum.

The Variable Neighborhood Search algorithms with the greedy agglomerative heuristic procedures proposed in [29,30] demonstrate better results than similar GAs. New randomly generated solutions (local minima) are used to form a randomized neighborhood around the best-achieved solution. This randomized approach provides some neighborhood variety.

The idea of this research is to use GAs with greedy agglomerative and other crossover operators in combination with the new mutation procedures which apply the same algorithm as the crossover procedure to the mutated solution and a randomly generated solution, and thus provide the population diversity.

2 New Crossover-Like Mutation Operator in a One-Point Crossover Genetic Algorithm

The GA in [14] uses the roulette wheel selection without any elitism (i.e., equal probabilities of selecting each of the individuals) and a simple one-point crossover procedure for the chromosomes. This algorithm uses the mutation procedure based on (2) with mutation probability 0.01. In our experiments below, we replaced this mutation procedure with the following algorithm:

Algorithm 2. k-Crossover-like mutation procedure with a one-point crossover

```
Step 1: Generating a random initial solution $S = \{X_1, ..., X_k\}$;
Step 2: Application of Algorithm 1 to $S$ for obtaining local optimum $S$;
Step 3: Applying the simple one-point crossover procedure to the mutated individual $S'$ from the population and $S$ for obtaining the new solution $S''$;
Step 4: Application of Algorithm 1 to $S''$ for obtaining local optimum $S''$;
Step 5: If $F(S'') < F(S')$ then $S' \leftarrow S''$.
```

This new procedure is used with probability equal to 1 after each crossover operator. In our experiments, the population size $N_{POP} = 20$. The results of running the original algorithm described in [14] and its version with Algorithm 2 as the mutation operator are shown in Table 1 and Fig. 1. Our experiments show that the new mutation procedure is faster and more effective.

Table 1. Computational results for Mopsi-Joensuu data set [31] (6014 two-dimensional data vectors), 300 clusters, time limitation 180 s.

GA generations	Result with the original mutation (2)	Result with the mutation (Algorithm 2)	Ordinary k-means in a multi-start mode
10	1697.29	1667.95	1859.06
20	1682.37	1664.78	
50	1679.58	1664.78	
150	1664.81	1664.78	
200	1664.78	1664.78	

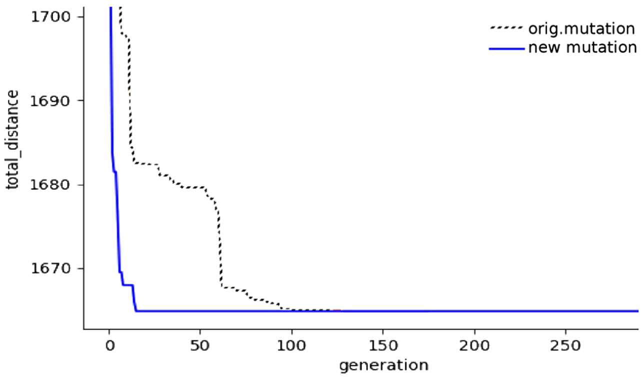

Fig. 1. Two mutation strategies in a one-point crossover GA

3 Known Clustering Algorithms of the Greedy Heuristic Method

The greedy agglomerative heuristic procedure for location problems can be described as an algorithm with two steps. The first step is combining two known ("parent") solutions (individuals) into one invalid intermediate solution with an excessive number of centroids. At the second step, the algorithm eliminates centroids in each iteration so that the removal of the centroid gives us the least significant increase in the value of the objective function (1) [11,16]:

Algorithm 3. Basic Greedy Agglomerative Heuristic Procedure

```
Require: needed number of clusters $k$, initial solution $S = \{X_1, ..., X_K\}$, $|S| = K$, $k < K$.
  Step 1: Improve $S$ with Algorithm 1 or other LSA.
  while $K > k$
    for all $i' \in \{\overline{1, K}\}$
      Step 2: Assign $S' \leftarrow S \setminus \{X'_i\}$. Calculate $F'_{i'} \leftarrow F(S')$ where $F(.)$ is the objective
      function value, (1) for the k-means problem.
    end for
    Step 3: Select a subset $S_{elim}$ of $n_{elim}$ centers, $S_{elim} \subset S$, $|S_{elim}| = n_{elim}$, with the
    minimal values of corresponding variables $F'_{i'}$. Here, $n_{elim} = \max\{1, 0.2(|S| - k)\}$.
    Step 4: Obtain new solution $S \leftarrow S \setminus S_{elim}$; $K \leftarrow K - 1$, and run an LSA.
  end while
```

Algorithms 4–5 are known heuristic procedures [11,29,32], which modify some given solution based on the second known solution (see Algorithm 3).

Algorithm 4. Greedy Procedure #1

```
Require: Two solutions (sets of centroids) $S' = \{X'_1, ..., X'_k\}$ and $S'' =$
  $\{X''_1, ..., X''_k\}$.
  for all $i' \in \{\overline{1, k}\}$
    Step 1: Merge $S'$ and one item of the set $S''$: $S \leftarrow S \cup \{X''_{i'}\}$.
    Step 2: Run Algorithm 3 with the initial solution $S$ and save the obtained result .
  end for
  Return the best of the solutions obtained on Step 2.
```

A simpler algorithm below combines the full "parent" solutions.

Algorithm 5. Greedy Procedure #2

```
  Combine sets $S \leftarrow S' \cup S''$, and run Algorithm 3 with the initial solution $S$.
```

These algorithms can be used in various global search strategies as their parts. Sets of solutions derived ("children") from the solution S' formed by combining its items with the items of some solution S'' and running Algorithm 1 are used as the neighborhoods in which a solution is searched. Thus, the second solution S'' is a parameter of the neighborhood selected randomly (randomized) [32].

4 GAs with Greedy Agglomerative Heuristic Procedures for the p-Median and k-Means Problems

The basic genetic algorithm for the k-means problem [7,26] can be described as follows:

Algorithm 6. GA with real-number alphabet for the k-means problem [18,25]

```
Require: Initial population size $N_{POP}$.
  Step 1: Select $N_{POP}$ initial solutions $S_1, ..., S_{N_{POP}}$ where $|S_i| = k$, and $\{S_1, ..., S_{N_{POP}}\}$ is a randomly chosen subset of the data vectors set. Improve each initial solution with Algorithm 1 and save corresponding obtained values of the objective function (1) as variables $f_k \leftarrow F(S_k),\ k = \overline{1, N_{POP}}$.
  loop
    Step 2: If the STOP condition is satisfied then STOP; return solution $S_{i^*}$, $i^* \in \{\overline{1, N_{POP}}\}$ with minimal value of $f_{i^*}$.
    Step 3: Randomly choose the two indexes $k_1$, $k_2 \in \{\overline{1, N_{POP}}\}$, $k_1 \neq k_2$.
    Step 4: run the crossover operator : $S_C \leftarrow Crossover(S_{k_1}, S_{k_2})$.
    Step 5: run the mutation operator : $S_C \leftarrow Mutation(S_C)$.
    Step 6: Run a selection procedure to change the population set.
  end loop
```

We used such a tournament selection on Step 6:

Algorithm 7. Tournament selection

```
  Randomly choose two indexes $k_4, k_5 \in \{\overline{1, N_{POP}}\}$, $k_4 \neq k_5$; if $f_{k_4} > f_{k_5}$ then $S_{k_4} \leftarrow S_C$, $f_{k_4} \leftarrow F(S_C)$ else $S_{k_5} \leftarrow S_C$, $f_{k_5} \leftarrow F(S_C)$.
```

Other selection methods do not significantly improve the result [11,16,21]. GAs with greedy agglomerative crossover can be described as follows [16,21]:

Algorithm 8. GA with greedy heuristic for the p-median problem and k-means problem (modifications GA-FULL, GA-ONE, and GA-MIX)

```
Step 1. Assign $N_{iter} \leftarrow 0$; select a set of the initial solutions $\{S_1, ..., S_{N_{POP}}\} \subset \{A_i | i = \overline{1, N}\}$, $|S_i| = k$. Improve each initial solution with Algorithm 1 and save the obtained values of the objective function (1) as variables $f_k \leftarrow F(S_k)$, $k = \overline{1, N_{POP}}$. We used initial populations with $N_{POP} = 5$.
loop
  Step 2: If STOP condition is satisfied then STOP; return solution $S_{i^*}$, $i^* \in \{\overline{1, N_{POP}}\}$ with minimal value of $f_{i^*}$ else adjust the population size : $N_{iter} \leftarrow N_{iter} + 1$; $N_{POP} \leftarrow \max\{N_{POP}, \lceil\sqrt{1 + N_{iter}}\rceil\}$; if $N_{POP}$ has changed, then initialize the new individual $S_{N_{POP}}$ as described in Step 1.
  Step 3: Randomly choose two indexes $k_1, k_2 \in \{\overline{1, N_{POP}}\}$.
  Step 4: Run Algorithm 4 (for GA-ONE modification) or Algorithm 5 (for GA-FULL modification) with "parent" solutions $S_{k_1}$ and $S_{k_2}$. For the GA-MIX modification, Algorithms 4 or 5 are chosen randomly with equal probabilities. Obtain new solution $S_C$.
  Step 5: $S_C \leftarrow Mutation(S_C)$. By default, no mutation procedure is used.
  Step 6: Run Algorithm 7.
end loop
```

This algorithm uses a dynamically growing population [16,17]. In our new version of Step 5, the Crossover-like mutation operator is as follows.

Algorithm 9. Crossover-like mutation operator (Step 5 of Algorithm 8, its modifications GA-FULL-MUT, GA-ONE-MUT, and GA-MIX-MUT)

```
Run the ALA algorithm (Algorithm 1) for a randomly chosen initial solution to get solution $S'$.
Run Algorithm 4 (for GA-ONE modification) or Algorithm 5 (for GA-FULL modification) with "parent" solutions $S_c$ and $S''$. Obtain new solution $S'_C$.
**If** $F(S'_C) < F(S_C)$, **then** $S_C \leftarrow S'_C$.
```

Our computational experiments (the next Section) show that new GAs with Algorithm 9 as the mutation operator are able to outperform both the original GAs with greedy agglomerative crossover operator (Algorithm 8) and the Variable Neighborhood Search with randomized neighborhoods (k-GH-VNS1 k-GH-VNS2, k-GH-VNS2) in some practically important problems.

5 Computational Experiments

We used data sets from the UCI (Machine Learning Repository) and the Clustering Basic Benchmark repositories [31,33] and the results of the non-destructive tests of prefabricated production batches of electronic radio components conducted in a specialized test center of JSC "TTC - NPO PM" used for the spacecraft equipment manufacturing [34]. The problem here is to divide a given mixed lot of radio components into clusters of similar devices manufactured from the same raw materials as a single homogeneous production batch. The test system consisted of Intel Core 2 Duo E8400CPU, 16 GB RAM. For all data sets, 30 attempts were made to run each of the algorithms. The j-means and k-means algorithms were launched in a multi-start mode [29,32].

Our new modifications (GA-xxx-MUT, Algorithm 8) of three GAs (Tables 2, 3) were compared to other known algorithms, such as corresponding known GAs without mutation (GA-FULL, GA-ONE, GA-MIX, see [25,30]). Variable Neighborhood Search with randomized neighborhoods (k-GH-VNS1, k-GH-VNS2, k-GH-VNS2, see [29,32]), their combinations with the j-means algorithm (j-means-GH-VNSx, see [29]), known GAs with the greedy agglomerative crossover procedures (GA-FULL, GA-ONE, GA-MIX modifications, see Algorithm 8), and other known algorithms (j-means and k-means, see [29,32]) in a multi-start mode.

The best-achieved values of the objective function (1) (its minimum value, mean value, and standard deviation) are underlined; the best values of new algorithms are given in a bold font, the best values of the known algorithms are given in italic. The results of the best of new algorithms (a sample of 30 results) were compared with the best of known tested algorithms (also 30 results) to prove the statistical significance of the advantage or disadvantage of new algorithms. We used the Mann-Whitney U-test and the t-test (significance level 0.01 for both tests).

Table 2. Computational experiment results for various data sets

Algorithm	Objective function (1) value			
	Min	Max	Average	Std.Dev
Europe data set (169309 data vectors of dimensionality 2) 30 clusters, 4 h				
j-means	7.51477E + 12	7.60536E + 12	7.56092E + 12	29.764E + 9
k-means	7.54811E + 12	7.57894E + 12	7.56331E + 12	13.560E + 9
k-GH-VNS1	*7.49180E + 12*	7.49201E + 12	*7.49185E + 12*	*0.073E + 9*
k-GH-VNS2	7.49488E + 12	7.52282E + 12	7.50082E + 12	9.989E + 9
k-GH-VNS3	7.49180E + 12	7.51326E + 12	7.49976E + 12	9.459E + 9
j-means-GH-VNS1	7.49180E + 12	7.49211E + 12	7.49185E + 12	0.112E + 9
j-means-GH-VNS2	7.49187E + 12	7.51455E + 12	7.4962E + 12	8.213E + 9
GA-FULL-MUT*	7.49293E + 12	7.49528E + 12	7.49417E + 12	0.934E + 9
GA-MIX-MUT*	7.49177E + 12	7.49211E + 12	7.49186E + 12	0.117E + 9
GA-ONE-MUT*↑⇑	**7.49177E + 12**	7.49188E + 12	**7.49182E + 12**	**0.042E + 9**
Testing results of the integrated circuits 5514BC1T2-9A5 (91 data vectors of dimensionality 173), grouping into 10 homogeneous batches (clusters), 2 min				
j-means	7 060.45	7 085.67	7 073.55	8.5951
k-means	7 046.33	7 070.83	7 060.11	8.8727
k-GH-VNS1	7 001.12	7 009.53	7 004.48	4.3453
k-GH-VNS2	7 001.12	7 010.59	7 002.26	2.9880
k-GH-VNS3	7 001.12	7 009.53	7 003.01	3.1694
j-means-GH-VNS1	*7 001.12*	7 001.12	*7 001.12*	*0.0000*
j-means-GH-VNS2	7 001.12	7 011.94	7 003.88	4.4990
GA-FULL-MUT*	7 001.12	7 001.27	7 001.24	0.0559
GA-MIX-MUT*↕⇕	**7 001.12**	7 001.12	**7 001.12**	**0.0000**
GA-ONE-MUT*↕⇕	7 001.12	7 001.12	7 001.12	0.0000
Ionosphere data set (351 data vectors of dimensionality 35, 10 clusters,1 min, Mahalanobis distance metric [35]				
k-means	9 253.2467	9 304.2923	9 275.3296	13.5569
k-GH-VNS1	9 083.6662	9 153.0192	9 121.0728	20.6875
k-GH-VNS2	9 085.8065	9 144.3779	9 112.2959	14.6803
k-GH-VNS3	9 090.5465	9 128.4111	9 109.8492	10.2740
GA-FULL	9 117.5695	9 175.1517	9 142.8457	15.3522
GA-FULL-MUT*	9 098.8748	9 157.0265	9 136.6556	16.1343
GA-MIX	9 095.1540	9 141.6417	9 114.1280	13.0638
GA-MIX-MUT*	9 102.8695	9 138.2243	9 113.4571	**8.6361**
GA-ONE	*9 078.6460*	9 115.9342	*9 099.7687*	*10.1104*
GA-ONE-MUT*↕⇕	**9 073.1919**	9 120.1842	*9 101.6286*	*12.8542*

Note: "*": new algorithm; "↑", "⇑": the advantage of the best of new algorithms over known algorithms is statistically significant ("↑" for t-test and "⇑" for Mann–Whitney U test), "↓", "⇓": the disadvantage of the best of new algorithms over known algorithms is statistically significant; "↕", "⇕": the advantage or disadvantage is statistically insignificant.

Table 3. Computational experiment results for various data sets

Algorithm	Objective function (1) value			
	Min	Max	Average	Std.Dev
Results of testing the integrated circuits 5514BC1T2-9A5 (91 data vectors of dimensionality 173), grouping into 10 homogeneous batches (clusters), 2 min, Mahalanobis distance [35]				
k-means	7 289.7935	7 289.8545	7 289.8296	0.0153
k-GH-VNS1	7 289.7366	7 289.8024	7 289.7648	0.0147
k-GH-VNS2	7 289.7472	7 289.8021	7 289.7792	0.0153
GA-FULL	7 289.7474	7 289.8134	7 289.7742	0.0175
GA-FULL-MUT*	**7 289.7062**	7 289.7754	7 289.7508	0.0181
GA-MIX	7 289.7319	7 289.7771	7 289.7501	*0.0133*
GA-MIX-MUT*	7 289.7227	7 289.7772	7 289.7496	**0.0149**
GA-ONE	*7 289.7228*	7 289.7796	*7 289.7494*	0.0159
GA-ONE-MUT*↕⇕	7 289.7147	7 289.7752	**7 289.7466**	0.0165
Results of testing the integrated circuits 5514BC1T2-9A5 (1234 data vectors of dimensionality 157), grouping into 10 homogeneous batches (clusters), 2 min				
j-means	43 841.97	43 843.51	43 842.59	0.4487
k-means	43 842.10	43 844.66	43 843.38	0.8346
k-GH-VNS1	43 841.97	43 844.18	43 842.34	0.9000
k-GH-VNS2	43 841.97	43 844.18	43 843.46	1.0817
k-GH-VNS3	43 841.97	43 842.10	43 841.99	0.0424
j-means-GH-VNS1	43 841.97	43 841.97	*43 841.97*	*0.0000*
j-means-GH-VNS2	43 841.97	43 844.18	43 842.19	0.6971
GA-FULL-MUT*	43 841.97	45 009.09	44 620.29	569.14
GA-MIX-MUT*	43 841.97	45 009.09	44 542.31	591.74
GA-ONE-MUT*↓⇓	**43 841.97**	45 009.09	**44 363.83**	**583.63**

6 Conclusion

We proposed a new approach to the design of Genetic Algorithms for the k-means problem with real-number (centroid-based) chromosome encoding, where the same procedure is used as both crossover and the mutation operators. Our experiments show that the GAs with one-point and greedy agglomerative crossover operators built in accordance with this idea outperform the algorithms without any mutation procedure and algorithms with the uniform random mutation by the obtained objective function value (SSE). In further research, our new approach can be applied to other problems such as p-median with various distance measures, k-medoids, mix probability distribution separation, etc. The efficiency of the GAs with greedy agglomerative crossover operators and Variable

Neighborhood Search algorithms with the randomized neighborhoods formed by greedy agglomerative procedures give us a reasonable hope for the successful application of our new idea for such problems.

References

1. Farahani, R., Hekmatfar, M.: Facility Location: Concepts, Models, Algorithms and Case Studies. Contributions to Management Science. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-7908-2151-2
2. Shirkhorshidi, A.S., Aghabozorgi, S., Wah, T.Y.: A comparison study on similarity and dissimilarity measures in clustering continuous data. PLoS ONE **10**(12) (2015). https://doi.org/10.1371/journal.pone.0144059
3. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982). https://doi.org/10.1109/TIT.1982.1056489
4. MacQueen, J. B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297 (1967)
5. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**(336), 846–850 (1971). https://doi.org/10.1080/01621459.1971.10482356
6. Hruschka, E., Campello, R., Freitas, A., de Carvalho, A.: A survey of evolutionary algorithms for clustering. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **39**, 133–155 (2009). https://doi.org/10.1109/TSMCC.2008.2007252
7. Freitas, A.A.: A review of evolutionary algorithms for data mining. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 371–400. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-09823-419
8. Zeebaree, D.Q., Haron, H., Abdulazeez, A.M. and Zeebaree, S.R.M.: Combination of K-means clustering with genetic algorithm: a review. Int. J. Appl. Eng. Res. **12**(24), 14238–14245 (2017). https://www.ripublication.com/ijaer17/ijaerv12n24_35.pdf
9. Hosage, C.M., Goodchild, M.F.: Discrete space location-allocation solutions from genetic algorithms. Ann. Oper. Res. J. **6**, 35–46 (1986). https://doi.org/10.1007/bf02027381
10. Bozkaya, B., Zhang, J., Erkut, E.: A genetic algorithm for the p-median problem. In: Drezner, Z., Hamacher, H. (eds.) Facility Location: Applications and Theory. Springer, Heidelberg (2002)
11. Alp, O., Erkut, E., Drezner, Z.: An efficient genetic algorithm for the p-median problem. Ann. Oper. Res. **122**, 21–42 (2003). https://doi.org/10.1023/A:1026130003508
12. Eremeev, A.V.: Genetic algorithm with tournament selection as a local search method. Discrete Anal. Oper. Res. **19**(2), 41–53 (2012)
13. Krishna, K., Murty, M.M.: Genetic K-Means algorithm. IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) **29**(3), 433–439 (1999). https://doi.org/10.1109/3477.764879
14. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern Recogn. J. **33**(9), 1455–1465 (2000). https://doi.org/10.1016/S0031-3203(99)00137-5
15. Neema, M.N., Maniruzzaman, K.M., Ohgai, A.: New genetic algorithms based approaches to continuous p-median problem. Netw. Spat. Econ. **11**, 83–99 (2011). https://doi.org/10.1007/s11067-008-9084-5

16. Kazakovtsev, L.A., Antamoshkin, A.N.: Genetic algorithm with fast greedy heuristic for clustering and location problems. Informatica **38**(3), 229–240 (2014). http://www.informatica.si/index.php/informatica/article/view/704/574
17. Kwedlo, W., Iwanowicz, P.: Using genetic algorithm for selection of initial cluster centers for the k-means method. In: ICAISC 2010: Artificial Intelligence and Soft Computing, pp. 165–172 (2010). https://doi.org/10.1007/978-3-642-13232-2_20
18. Kim, K., Ahn, H.: A recommender system using GA K-means clustering in an online shopping market. Expert Syst. Appl. **34**(2), 1200–1209 (2008)
19. He, Z., Yu, C.: Clustering stability-based evolutionary k-means. Soft Comput. **23**(1), 305–321 (2018). https://doi.org/10.1007/s00500-018-3280-0
20. Naldi, M.C., Campello, R.J.G.B., Hruschka, E.R., Carvalho, A.C.P.L.F.: Efficiency issues of evolutionary k-means. Appl. Soft Comput. **11**(2), 1938–1952 (2011). https://doi.org/10.1016/j.asoc.2010.06.010
21. Graña, M., López-Guede, J.M., Etxaniz, O., Herrero, Á., Quintián, H., Corchado, E. (eds.): SOCO/CISIS/ICEUTE -2016. AISC, vol. 527. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-47364-2
22. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987). https://doi.org/10.1016/0377-0427(87)90125-7
23. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 224–227 (1979). https://doi.org/10.1109/TPAMI.1979.4766909
24. Pelleg, D., Moore A.: X-means: extending K-means with efficient estimation of the number of clusters. In: Proceedings of the 17th International conference on Machine Learning, pp. 727–734 (2000)
25. Kazakovtsev, L.A., Antamoshkin, A.N.: Greedy heuristic method for location problems. Vestnik SibGAU **16**(2), 317–325 (2015). https://cyberleninka.ru/article/n/greedy-heuristic-method-for-location-problems
26. Kwedlo, W.: A clustering method combining differential evolution with the k-means algorithm. Pattern Recogn. Lett. **32**(12), 1613–1621 (2011). https://doi.org/10.1016/j.patrec.2011.05.010
27. Chang, D.-X., Zhang, X.-D., Zheng, C.-W.: A genetic algorithm with gene rearrangement for k-means clustering. Pattern Recogn. **42**(7), 1210–1222 (2009). https://doi.org/10.1016/j.patcog.2008.11.006
28. Brimberg, J., Drezner, Z., Mladenovic, N., Salhi, S.: A new local search for continuous location problems. Eur. J. Oper. Res. **232**(2), 256–265 (2014)
29. V I Orlov, V.I., Kazakovtsev, L.A., Rozhnov, I.P., Popov, N.A., Fedosov, V.V.: Variable neighborhood search algorithm for k-means clustering. IOP Conf. Ser. Mater. Sci. Eng. 450, 022035 (2018). https://doi.org/10.1088/1757-899X/450/2/022035
30. Kazakovtsev, L., Stashkov, D., Gudyma, M., Kazakovtsev, V.: Algorithms with greedy heuristic procedures for mixture probability distribution separation. Yugoslav J. Oper. Res. **29**(1), 51–67 (2019). https://doi.org/10.2298/YJOR1711
31. Clustering basic benchmark. http://cs.joensuu.fi/sipu/datasets
32. Orlov, V.I., Rozhnov, I.P., Kazakovtsev, L.A., Lapunova, E.V.: An approach to the development of clustering algorithms with a combined use of the variable neighborhood search and greedy heuristic method. IOP Conf. Ser. **1399** (2019). https://doi.org/10.1088/1742-6596/1399/3/033049
33. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

34. Rozhnov, I., Orlov, V. Kazakovtsev, L.: Ensembles of clustering algorithms for problem of detection of homogeneous production batches of semiconductor devices. In: CEUR Workshop Proceedings OPTA-SCL 2018. Proceedings of the School-Seminar on Optimization Problems and their Applications. CEUR-WS 2098, pp. 338–348 (2018). http://ceur-ws.org/Vol-2098/paper29.pdf
35. McLachlan, G.J.: Mahalanobis distance. Resonance **4**(20), 1–26 (1999). https://doi.org/10.1007/BF02834632