View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

FACTA UNIVERSITATIS (NIŠ)
SER. MATH. INFORM. Vol. 34, No 5 (2019), 957–972
https://doi.org/10.22190/FUMI1905957R

VNS-BASED ALGORITHMS FOR THE CENTROID-BASED CLUSTERING PROBLEM *

Ivan P. Rozhnov, Victor I. Orlov and Lev A. Kazakovtsev

© 2019 by University of Niš, Serbia | Creative Commons Licence: CC BY-NC-ND

Abstract. The k-means algorithm with the corresponding problem formulation is one of the first methods that researchers use when solving a new automatic grouping (clustering) problem. Its improvement, modification and combination with other algorithms are described in the works of many researchers. In this research, we propose new algorithms of the Greedy Heuristic Method, which use an idea of the search in variable neighborhoods for solving the classical cluster analysis problem, and allows us to obtain a more accurate and stable result of solving in comparison with the known algorithms. Our computational experiments show that the new algorithms allow us to obtain results with better values of the objective function value (sum of squared distances) in comparison with classical algorithms such as k-means, j-means and genetic algorithms on various practically important datasets. In addition, we present the first results for the GPU realization of the Greedy Heuristic Method.

Keywords. Greedy Heuristic; clustering problem; GPU; k-Means Problem; variable neighborhoods.

1. Introduction

The use of the automatic grouping (clustering) systems is becoming increasingly common due to the expansion of application areas of data analysis problems such as image recognition, diagnostic problems in medicine, marketing research, Internet traffic research, etc. [1-3].

The idea of the k-means algorithm was proposed in 1956 by Steinhaus [4], and the algorithm was developed by Lloyd in 1957, although his work [5] was published only in 1982. After the article of McQueen [6], the algorithm became known as the k-means algorithm, or the Lloyd algorithm. Since then, the k-means algorithm, its

Received May 19, 2019; accepted August 13, 2019
2010 *Mathematics Subject Classification.* Primary 90C57; Secondary 90C27, 90C09
*Results were obtained in the framework of the state task 2.5527.2017/8.9 of the Ministry of Education and Science of the Russian Federation

improvement, modification, and combination with other algorithms have become the topic of many researchers [7].

The algorithm includes only two alternating steps: splitting a set of objects into groups (clusters) around known centers (the object belongs to the group which has center closest to this object) and redefining the centers of the groups. In the case of the classical k-means problem, the cluster centers are usually called centroids.

An important issue of the k-means algorithm is the choice of initial centers, which is often the subject of a special research [8-12]. Busare and Bansode [8] describe the k-means algorithm in combination with an improved pillar algorithm, which is efficient for selecting initial centers, however, its outlier problems lead to a decrease in performance. The authors have solved these problems. Wang et al. [9] used the Huffman tree to build the dissimilarity matrix and select initial centers.

For weighted multidimensional data, Mahmud et al. [10] used the choice of initial centers by a heuristic method, which takes into account the weight of each data attribute. Abdul Nazir and Sebastian [11] describe an improved k-means algorithm, which includes special methods for determining initial centers and assigning points to clusters. However, any technique of selecting initial values does not turn the k-means algorithm into a global search algorithm. For the large multidimensional datasets, its result are very far from the true minimum of the problem.

The J-means algorithm, developed by Hansen and Mladenovic [12], is considered as one of the most efficient and accurate algorithms for this problem, as well as for the p-median problem. The algorithm replaces the centers with one of the data vectors. If such replacement leads to the objective function value decrease, it continues the search using the standard k-means procedure.

The commonality of clustering problems based on the model of k-means and similar (searched parameters of such problems are the coordinates of cluster centers) with problems based on the model of mixture probability distribution separation [13], in which the parameters of the problem are the unknown parameters of distributions (including mathematical expectations which can be considered as cluster centers), supplemented by a priori probabilities of distributions. In addition, the most popular algorithms for solving such problems, k-means and the EM-algorithm [13], are similar in their structures: both are procedures with two alternating steps of the dividing of objects into clusters and location of the centers. In both types of problems, the objective functions are multi-extremal. This allows us to use similar methods for increasing the accuracy and stability of the solutions obtained by algorithms for both types of problems.

In the works of Stashkov et al. [14, 15], authors consider the use of the VNS algorithm (Variable Neighborhood Search) as an extended local search method in combination with an EM algorithm. The essence of the variable neighborhood approach [16] is that for some intermediate solution, we determine a set of neighborhoods of this solution. From this set, we select one neighborhood, in which, using the appropriate local search algorithm, the algorithm searches for a solution with the better value of the objective function. If such a solution has been found, the intermediate solution is replaced by this new solution, and the algorithm goes

on with the search in the same neighborhood. If such a solution cannot be found, algorithm selects a new neighborhood.

The idea of a greedy agglomerative heuristic procedure for the k-means problem in combination with VNS algorithms was first proposed in [7]. The idea of this paper is to apply improved versions of the Greedy Heuristic Method algorithms [17] using the idea of search in alternating neighborhoods to the k-means problem for obtaining the most accurate (by the value of the objective function) and a stable result. The results of our computational experiments are given in comparison with both classical algorithms and other algorithms of the Greedy Heuristic Method, including on those problems where the Genetic Algorithms of the Greedy Heuristic Method have proven themselves well.

2. The k-Means Problem and Algorithm

The k-means problem [18] consists in finding such set of k cluster centers $X_1 ... X_k$ in d-dimensional space so that the sum of squared distances from them to the given points A_i (SSE, sum of squared errors) is minimal.

$$\arg\min F(X_1, ..., X_k) = \sum_{i=1}^{N} \min_{j \in \{1,k\}} \|X_j - A_i\|^2. \tag{2.1}$$

The algorithm of the same name sequentially improves a known solution, allowing down to a local minimum of (2.1). However, in the strict sense, it is not a local search algorithm of a continuous optimization problem, since the search for a new solution is not carried out in an ε-neighborhood. This is a simple and fast algorithm applicable to the widest class of problems. The algorithm has some limitations, in particular, we need to pre-set the number of groups k into which objects are clustered. The result is highly dependent on the initial decision, usually chosen randomly.

Algorithm 1 k-means (ALA procedure: alternating location and allocation)
Required : data vectors $A_1 ... A_N, k$ initial cluster centers $X_1 ... X_k$
do
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.
2 : Calculate new center X_i for each of clusters.
while Steps 1-2 lead to any modifications.

The aim of our study is to improve the accuracy of the result of solving the k-means problem (by the value of the objective function) and the stability of the result (1), for a fixed, limited time.

3. Clustering Algorithms of the Greedy Heuristic Method

A greedy agglomerative heuristic procedure for the problem of k-means and similar problems [19] consists of two steps. Such procedures require two known ("parent") solutions to the problem (the first of which, for example, is the best achieved solution), represented by sets of cluster centers S.

First, the sets of parent solutions are merged. We obtain an intermediate infeasible solution with an excessive number of clusters.

Then, the number of centers is gradually reduced. Each time, the algorithm cuts off such a center, that the removal of this center gives the least significant deterioration in the value of the objective function (1).

The algorithm of the basic greedy agglomerative heuristic procedure is as follows:

Algorithm 2 Basic Greedy Agglomerative Heuristic Procedure
Required : initial number of clusters K, required number of clusters $k < K$
initial solution $S, |S| = K$.
1: Improve the solution S with Algorithm 1 (if possible).
while $K \neq k$ **do**
for each $i' \in \{\overline{1, K}\}$ **do**
2.1: $S' = S \backslash \{X_{i'}\}$.
2.2: Improve $S'$with Algorithm 1, having performed 1-3 of its iterations. Store the obtained value of (1) to $F'_{i'}$.
end do
3: $i'' = \arg\max_{i'=\overline{1,k}} F_{i'}$
4: Compose new solution $S'' = S \backslash \{X_{i'}\}$, and improve it with Algorithm 1.
end do

We have proposed new heuristic procedures that modify the known solution based on the second known solution.

Similar greedy agglomerative heuristic procedures were used as crossover operators in the evolutionary (genetic) algorithms of the Greedy Heuristic Method [17, 19]. In this paper, these procedures form neighborhoods that are used for modifying the known solution when searching in a search algorithm in alternating neighborhoods.

Algorithm 3 Greedy Procedure with Partial Merger $\neq 1$
Required: two "parent" sets (arrays) of cluster centers $S' = \{X'_1, ..., X'_k\}$ and $S'' = \{X''_1, ..., X''_k\}$
Calculate the objective function (1): $F^* = F(S')$;
Arrange the elements of S''in ascending order of values $F(S \cup \{X''_k\})$
for each $i' \in \{\overline{1, K}\}$**do**
1: Attach an element of $F(S \cup \{X''_k\})$,
2: Run Algorithm 2 with initial solution S. Save the obtained set of cluster centers S_i and corresponding value F_i of the objective function (1). If $F_i < F^*$ then $S' = S$.
end do
3. Return the best solution obtained in Step 2.

A simpler version of the greedy agglomerative procedure, but demanding more computational resources is presented below. Note that, unlike Algorithm 3, this algorithm was proposed earlier [17, 19] as a crossing-over operator for a Genetic Algorithm with a greedy agglomerative heuristic procedure.

Algorithm 4 Greedy Procedure with Full Merger
Required: see Algorithm 3.

1: Combine sets $S = S' \cup S''$.
2: Run Algorithm 2 with initial solution S.

Finally, an compromise variant is possible, in which the sets are partially combined: the first parent set being taken completely, with a random number of elements chosen randomly from the second set [17, 20].

Algorithm 5 Greedy Procedure with Partial Merger $\neq 1$
Required: see Algorithm 3.

1: Generate randomly $r' \in [0; 1)$. Calculate $r = [(k/2 - 2) \cdot r'^2] + 2$.
Here, [.] is the integer part.
2 **for** $i = 1$ **to** $k - r$ **do**
2.1: Form a randomly chosen subset $S'''$of S'' of cardinality r.
Combine sets $S = S' \cup S'''$.
2.2: Run Algorithm 2 with the initial solution S.
end do
3: Return the best solution obtained in Step 2.2.

These heuristic procedures, which are (not in the strict sense) algorithms of local search in the neighborhood of known ("parent") solution represented by the set S can be embedded in various global search strategies. These procedures can be used for composing new solutions which are derived from solution S' ("child" solutions of S by combining S with other solution S''. Thus, greedy procedures form the neighborhoods of S, and S'' is a parameter of such neighborhoods.

The Greedy Heuristic Method [19] provides, as one of the options for organizing a local search, the use of search algorithms with variable neighborhoods (VNS algorithms) embedded in the greedy agglomerative procedures [21-23]. A review of current local search methods based on the idea of variable neighborhoods and approaches to combining them with other metaheuristics is given in [24]. In this paper, we use the greedy agglomerative heuristic procedures embedded in the VNS algorithm.

Algorithms 3-5 perform the search in the neighborhood formed by extension of a known intermediate solution S' by elements of another solution S'' with the subsequent removal of the extra cluster centers with the use of the greedy agglomerative heuristic procedure. Thus, these algorithms search in some neighborhoods of solution S', and the second solution S'' is a parameter of this neighborhood, which is selected randomly (randomized). The idea of the proposed VNS-based algorithm for solving the k-means problem can be described as follows:

Algorithm 6 k-VNS (general description of the algorithm family) $\neq 1$

1: Run Algorithm 1 with randomly generated initial solution S.
2 $O = O_{start}$. Here, O_{start} is the index of the initial neighborhood type.
3: $i = 0, j = 0$.
while $j < j_{max}$ **and** stop conditions (time limitation) are not satisfied **do**
while $i < i_{max}$ **do**
4: Run Algorithm 1 from a randomly generated solution and obtain new solution S'.
repeat
5: Depending on the value of variable O(values equal to 1, 2 and 3 are allowed), run Algorithm 3, 4 or 5, correspondingly, with initial solutions S and S'. Thus, the neighborhood is determined by variables O and S'.
if the result of Step 5 is better than S **then**
store new result to $S, i = 0, j = 0$.
until Step 5 results in a solution which is worse than S
6: $i = i + 1$.
end do
7: $i = 0, j = j + 1, O = O + 1$; **if** $O > 3$ **then** $O = 1$.
end do

For this algorithm, values of two parameters are very important: i_{max} is the number of unsuccessful attempts to search in the current type of neighborhood, and j_{max} is the number of unsuccessful switching of neighborhood types. We used the values $i_{max} = \min\{2k,20\}, j_{max} = 2$.

In addition, parameter O_{start},which specifies the number of the initial type of neighborhood. plays very important role. The parameter determines the type of local minimum to which the solution is initially attracted. As shown by computational experiments, the initial type of the neighborhood largely determines the future behavior and the result of the algorithm. We performed our computational experiments with all its possible values. Depending on this value, the algorithms are designated below respectively k-VNS1, k-VNS2, k-VNS3.

Moreover, the method of obtaining the second solution S' in Step 4 is important, too. By default, the second solution contains the number of centers equal to the number of centers in solution S. We also used modifications of Algorithm 6, in which the number of centers in solution S' is chosen randomly from the set $\overline{\{2, 2|S|\}}$,where $|S|$ is the number of centers in the solution S. In this case, the algorithms are named k-VNS1-RND, k-VNS2-RND, k-VNS3-RND.

Note that for k-means problems, the j-means procedure is considered to be very efficient. However, the scope this procedure is limited to relatively small problems due to computational complexity. This procedure is reduced to replacing the centers with one of the data vectors. If such replacement is successful (from the point of view of the objective function) then algorithm continues the search with the standard k-means procedure [13, 19]. Thus, the j-means is a VNS algorithm.

In addition, for our computational experiments, we used the combined versions

of the VNS and J-Means algorithms, denoting them: J-Means-VNS1, J-Means-VNS2. In addition to three values of the neighborhood type O(as in Algorithm 6), these versions of algorithms use the 4^{th}possible value. If the value is equal to 1, then these algorithms use the J-means for local search, otherwise they run Algorithms 3-5. In algorithms J-Means-VNSx, Steps 5 and 7 are as follows: 5: Depending on the value of variable O (values 1, 2, 3 and 4 are allowed), run algorithm J-Means, Algorithms 3, 4, or 5, correspondingly, with initial solutions S and S'.
7: $i = 0, j = j + 1, O = O + 1$, **if** $O > 4$ **then** $O = 1$**.**

4. Computational Experiments

To test our new family of algorithms (k-VNSx), we used the classical data sets from the UCI Machine Learning Repository [25] and Clustering Datasets [26].

For our experiments (Tables 4.1-4.4), we used the computer system Depo X8Sti (Xeon X5650 2.67 GHz CPU, 12 GB of RAM). Some of the experiments were also conducted on a small system with the Atom N270 1.6 GHz CPU, 1 GB of RAM (the same results if the execution time increases 16-25 times). For all data sets, 30 attempts were done with each of the algorithms (J-Means, k-means, k-VNS1, k-VNS2, k–VNS3, k–VNS1–RND, k-VNS2-RND, k-VNS3-RND, J–Means–VNS1, J–Means–VNS2). Only the best results achieved in each attempt were recorded. For these results of each algorithm, the best and the worst, and the averaged values were calculated (Min, Max, Average). To estimate the stability of the results, we calculated the standard deviation (SD) of (1). The J-Means and k-means algorithms were launched in multi-start mode. On selected data sets, we performed calculations with various numbers of the required clusters (Tables 4.1, 4.4), the execution time of the algorithms (Tables 4.2, 4.4).

The best minimum and average values of the objective function for each of dataset and the minimum standard deviations are in italics (Tables 4.1-4.4).

The results of our computational experiments (Tables 4.1-4.4) showed that new VNS-based algorithm family allows obtaining (lower average value of the objective function and / or its standard deviation, a smaller spread of the achieved values) in comparison with the classical algorithms of J-Means and k-means (table 4.5). At this stage of our research, it is difficult to give a definite preference for any one of the versions of the VNS algorithm or its combined versions of J-Means-VNS.

The results of computational experiments (Tables 4.1-4.4) are summarized in Table 4.5, which shows the number of the best achieved values f the objective function for each of the solved problems (data sets from the UCI and Clustering Datasets repositories) among all clustering algorithms.

For testing purposes, we also used the results of non-destructive test tests of prefabricated batches of electrical and radio products, conducted in a specialized test center to complete the onboard equipment of spacecraft. The best values of the objective function (minimum value, mean value and standard deviation) are shown in bold italics (Tables 4.6-4.8).

Table 4.1: Computational experiments with dataset "ionosphere" (351 data vectors of dimensionality 35), 30 sec., 30 attempts

Algorithm	Min (the best)	Max	Average	Std.dev.
10 clusters				
J-Means	1 590.34	1 598.83	1 594.77	2.41
k-means	1 590.93	1 598.61	1 595.28	2.47
k-VNS1	**1 586.38**	1 586,65	**1 586.52**	**0.12**
k-VNS2	**1 586.38**	1 591.99	1 588.00	2.36
k-VNS3	**1 586.38**	1 591.99	1 587.24	1.57
J-Means-VNS1	**1 586.38**	1 586.63	**1 586.44**	**0.10**
J-Means-VNS2	**1 586.39**	1 586.63	**1 586.48**	**0.12**
20 clusters				
j-Means	1 282.18	1 299.13	1 291.92	4.83
k-means	1 286.30	1 310.54	1 301,98	5.81
k-VNS1	1 239.16	1 259.56	**1 246.39**	5.18
k-VNS2	1 243.94	1 263.11	1 252.26	4.96
k-VNS3	**1 238.53**	1 265.28	1 252.53	6.47
J-Means-VNS1	**1 236.21**	1 257.85	**1 245.97**	6.48
J-Means-VNS2	1 245.71	1 256.18	1 249.95	**3.09**

As seen from the results of our computational experiments, the new algorithms again gave more stable results.

In addition, we compared the results of performed computational experiments on data sets of semiconductor devices with the results of various modifications of the Genetic Algorithm [19] including the Genetic Algorithm with Greedy Heuristic. Comparative results are shown in Table 4.9. Here, we used the following abbreviations [19]: GA classical - genetic algorithm with random recombination, GH - greedy heuristic procedures in the multistart mode, GAGH FP - Genetic Algorithm with greedy heuristics and floating point alphabet, LS - local search, GAFS - genetic algorithm with recombination of fixed length subsets [27], IBC - Information Bottleneck Clustering.

As we see from Table 4.9, in some cases the objective function values achieved by of our new algorithms turn out to be significantly better and more stable than the results of the genetic algorithms. At the same time, for some problems, new algorithms are inferior to genetic algorithms, but not significantly. Nevertheless, new algorithms are. It should be noted that for medium-sized problems (up to about 10,000 data vectors, up to 100 clusters), new algorithms lose their advantage over genetic algorithms if we increase the time limitation significantly.

Table 4.2: Computational experiments with dataset "chess" (3196 Boolean data vectors of dimensionality 37), 30 clusters, 30 attempts

Algorithm	Min (the best)	Max	Average	Std.dev.
3 min.				
J-Means	8 021.22	8 102.13	8 060.24	21.05
k-means	7 989.20	8 038.81	8 019.24	13.68
k-VNS1	**7 960.82**	7 978.85	7 967.27	4.99
k-VNS2	**7 959.92**	7 989.01	7 974.27	8.61
k-VNS3	7 998.48	8 007.96	8 003.84	3.51
k-VNS1-RND	7 960.66	7 978.62	7 968.86	5.94
k-VNS2-RND	7 961.89	7 996.10	7 972.31	8.76
k-VNS3-RND	7 987.20	8 001.26	7 991.42	3.55
J-Means-VNS1	**7 958.25**	7 967.75	**7 961.82**	4.65
J-Means-VNS2	**7 959.03**	7 970.65	**7 963.63**	4.13
1 hour				
J-Means	7 997.43	8 031.05	8 014.72	10.71
k-means	7 970.88	8 005.28	7 990.12	9.31
k-VNS1	**7 958.26**	7 969.10	7 962.73	3.89
k-VNS2	**7 958.25**	7 961.61	**7 959.34**	1.21
k-VNS3	**7 958.26**	7 963.07	7 960.22	2.03
k-VNS1-RND	**7 958.24**	7 965.03	7 960.91	1.59
k-VNS2-RND	**7 958.24**	7 963.09	**7 959.57**	1.56
k-VNS3-RND	**7 958.24**	7 968.36	**7 959.49**	2.64
J-Means-VNS1	**7 958.25**	7 958.28	**7 958.26**	**0.02**
J-Means-VNS2	**7 958.25**	7 960.39	**7 958.68**	**0.85**

Table 4.3: Computational experiments with dataset "birch3" (100000 data vectors, 2-dimensional) 100 clusters, 2 hours, 30 attempts

Algorithm	Min (the best)	Max	Average	Std.dev.
J-Means	3.7622E+13	3.7965E+13	3.7772E+13	0.1162E+12
k-means	7.9247E+13	8.8740E+13	8.3160E+13	3.0881E+12
k-VNS1	**3.7254E+13**	3.7747E+13	3.7470E+13	0.1711E+12
k-VNS2	4.2138E+13	6.0108E+13	5.1898E+13	7.1913E+12
k-VNS3	**3.7253E+13**	3.7457E+13	**3.7375E+13**	0.0743E+12
k-VNS1-RND	**3.7254E+13**	3.7769E+13	3.7494E+13	0.1855E+12
k-VNS2-RND	3.8326E+13	4.6185E+13	4.0815E+13	2.5432E+12
k-VNS3-RND	3.7313E+13	3.7524E+13	3.7416E+13	**0.0618E+12**
J-Means-VNS1	**3.716E+13**	3.7181E+13	**3.7174E+13**	**0.0122E+12**
J-Means-VNS2	**3.7242E+13**	3.7456E+13	**3.7347E+13**	0.107E+12

Table 4.4: Computational experiments with dataset "KDDCUP04Bio" (145751 data vectors of dimensionality 74, normalized), 30 attempts

Algorithm	Min (the best)	Max	Average	Std.dev.
30 clust., 200 min.				
J-Means	6 280 406	6 288 774	6 283 271	4 767.4
k-means	6 310 843	6 429 357	6 370 635	63 853.5
k-VNS1	6 385 012	6 385 150	6 385 047	**51.3**
k-VNS2	6 385 196	6 430 515	6 418 326	17 204.4
k-VNS3	**6 267 808**	6 386 810	6 366 953	48 570.7
k-VNS1-RND	6 385 016	6 385 033	6 385 023	**6.2**
k-VNS2-RND	6 385 149	6 429 426	6 410 598	16 538.2
k-VNS3-RND	6 386 703	6 386 808	6 386 753	**50.4**
J-Means-VNS1	**6 267 205**	6 267 395	**6 267 300**	134.3
J-Means-VNS2	**6 267 217**	6 267 421	**6 267 319**	144.2
200 clust., 12 hrs				
J-Means	5 330 344	5 382 908	5 355 903	26 785.8
k-means	5 336 446	5 381 386	5 366 144	25 722.4
k-VNS1	**5 294 620**	5 307 828	**5 301 224**	**9 339.5**
k-VNS2	5 440 814	5 490 400	5 465 607	35 062.8
k-VNS3	No result			
k-VNS1-RND	**5 310 067**	5 340 849	5 325 458	21 765.7
k-VNS2-RND	5 368 527	5 399 695	5 384 111	22 039.6
k-VNS3-RND	No result			
J-Means-VNS1	5 430 120	5 446 222	5 438 171	11 385.8
J-Means-VNS2	5 500 410	5 508 985	5 504 697	**6 063.4**
2000 clust., 8 hrs				
J-Means	4 390 323	4 404 301	4 396 180	5 912.2
k-means	4 424 475	4 426 251	4 425 137	**786.5**
k-VNS1	4 358 583	4 362 786	4 360 886	2 130.1
k-VNS3	**4 311 992**	4 318 547	**4 315 658**	2 721.5

Table 4.5: The numbers of the achieved best values and best averaged values of the objective function and its standard deviation by each of the algorithms among all the algorithms

Algorithm	The best position by minimum (the best value of the obj. function)	The best position by std.dev.	The best position by min. and std.dev. simultaneously
J-Means	0	0	0
k-means	0	1	0
k-VNS1	5	3	2
k-VNS2	3	0	0
k-VNS3	6	1	0
k-VNS1-RND	3	1	0
k-VNS2-RND	1	0	0
k-VNS3-RND	1	3	0
J-Means-VNS1	6	3	3
J-Means-VNS2	5	4	2

Table 4.6: Computational experiments with testing data of semiconductor devices 3OT122A. (767 data vectors of dimensionality 13), 10 clusters, 1 minute, 30 attempts

Algorithm	Min (the best)	Max	Average	Std.dev.
J-Means	**772.66**	772.70	772.68	0.0191
k-means	**772.66**	772.66	**772.66**	**0.0000**
k-VNS1	**772.66**	772.66	**772.66**	**0.0000**
k-VNS2	**772.66**	772.66	**772.66**	**0.0000**
k-VNS3	**772.66**	772.66	**772.66**	**0.0000**
k-VNS1-RND	**772.66**	772.66	**772.66**	**0.0000**
k-VNS2-RND	**772.66**	772.66	**772.66**	**0.0000**
k-VNS3-RND	**772.66**	772.66	**772.66**	**0.0000**
J-Means-VNS1	**772.66**	772.66	**772.66**	**0.0000**
J-Means-VNS2	**772.66**	772.66	**772.66**	**0.0000**

Table 4.7: Computational experiments with testing data of semiconductor devices 5514BC1T2-9A5 (91 data vectors, dimensionality 173) 10 clusters, 2 minutes, 30 attempts

Algorithm	Min (the best)	Max	Average	Std.dev.
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-VNS1	**7 001.12**	7 009.53	7 004.48	4.3453
k-VNS2	**7 001.12**	7 010.59	**7 002.26**	**2.9880**
k-VNS3	**7 001.12**	7 009.53	7 003.01	**3.1694**
J-Means-VNS1	**7 001.12**	7 001.12	**7 001.12**	**0.0000**
J-Means-VNS2	**7 001.12**	7 011.94	7 003.88	4.4990

Table 4.8: Computational experiments with data of semiconductor devices 1526TL1 (1234 data vectors of dimensionality 157), 2 minutes, 30 attempts

Algorithm	Min (the best)	Max	Average	Std.dev.
5 clusters				
J-Means	**63 337.29**	63 337.56	**63 337.46**	0.1211
k-means	**63 337.29**	63 337.29	**63 337.29**	**0.0000**
VNS1	**63 337.47**	63 337.56	**63 337.55**	0.0280
VNS2	**63 337.56**	63 337.56	**63 337.56**	**0.0000**
VNS3	**63 337.56**	63 337.56	**63 337.56**	**0.0000**
VNS1-RND	**63 337.56**	63 337.56	**63 337.56**	**0.0000**
VNS2-RND	63 **337.56**	63 337.56	**63 337.56**	**0.0000**
VNS3-RND	**63 337.56**	63 337.56	**63 337.56**	**0.0000**
J-Means-VNS1	**63 337.56**	63 337.56	**63 337.56**	**0.0000**
J-Means-VNS2	**63 337.56**	63 337.56	**63 337.56**	**0.0000**
10 clusters				
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
VNS1	**43 841.97**	43 844.18	**43 842.34**	0.9000
VNS2	**43 841.97**	43 844.18	43 843.46	1.0817
VNS3	**43 841.97**	43 842.10	**43 841.99**	**0.0424**
J-Means-VNS1	**43 841.97**	43 841.97	**43 841.97**	**0.0000**
J-Means-VNS2	**43 841.97**	43 844.18	**43 842.19**	0.6971

Table 4.9: Computational experiments with testing data of semiconductor devices (10 clusters, 1 minute, 30 attempts)

Algorithm	Min (the best)	Max	Average	Std.dev.
1526TL1 (1234	data vectors,	dimensionality	157)	
J-Means	**43841.97**	43843.51	43842.59	0.4487
k-means	43842.10	43844.66	43843.38	0.8346
k-VNS1	**43841.97**	43844.18	43842.34	0.9000
k-VNS2	**43841.97**	43844.18	43843.46	1.0817
k-VNS3	**43841.97**	43842.10	**43841.99**	**0.0424**
J-Means-VNS1	**43841.97**	43841.97	**43841.97**	**0.0000**
J-Means-VNS2	**43841.97**	43844.18	43842.19	0.6971
GAGH+LS	43842.10	43845.73	43843.72	1.3199
GAGHFP.	**43841.98**	43844.18	43842.6	0.6762
GAFS	**43841.98**	43842.34	43842.10	0.0945
GA classical.	43842.10	43842.88	43842.44	0.2349
IBC	Noresult			
Determ.GH	45021.21	45021.21	45021.21	**0.0000**
2D522B (3711	data vectors,	dimensionality	10)	
J-Means	7719.98	7720.74	7720.36	1.0174
k-means	7718.57	7722.91	7720.74	2.8714
k-VNS1	7716.88	7717.18	7717.03	0.0738
k-VNS2	7722.32	7726.42	7724.37	1.8752
k-VNS3	7722.81	7725.22	7724.51	1.3946
J-Means-VNS1	7717.22	7721.40	7719.81	1.7851
J-Means-VNS2	7717.90	7720.14	7719.92	1.4016
GAGH+LS	**7714.13**	7715.50	**7714.61**	0.3837
GAGH FP.	**7714.15**	7714.77	**7714.66**	0.1954
GAFS	**7714.14**	7714.29	**7714.22**	**0.0612**
GA classical.	**7714.14**	7714.30	**7714.21**	**0.0678**
DeterministicGH	7902.21	7902.21	7902.21	**0.0000**
H5503XM1 (3701	data vectors,	dimensionality	229)	
J-Means	43675.96	43681.52	43678.74	1.4126
k-means	43675.90	43684.88	43679.77	2.8062
VNS1	**43671.89**	43671.89	**43671.89**	**0.0000**
VNS2	43672.24	43674.44	43673.34	1.0476
VNS3	43672.84	43675.76	43674.30	1.5916
J-Means-VNS1	**43671.89**	43671.89	**43671.89**	**0.0000**
J-Means-VNS2	43673.14	43675.56	43674.35	0.9162
GAGH+LS	43702.28	43766.87	43739.69	20.3107
GAGH FP.	43678.79	43693.63	43687.01	4.5961
GAFS	43708.14	43736.26	43716.26	8.4025
GA classical.	43703.31	43724.42	43715.80	6.1660
Deterministic GH	43830.25	43830.25	43830.25	**0.0000**

5. Conclusions

The results of our computational experiments showed that the new search VNS-based algorithms allows us to in comparison with known algorithms.

For large-scale problems, as the number of clusters grows and the sample size increases, the comparative efficiency of the new algorithm increases, new VNS-based algorithms have an advantage. Thus, the arsenal of high-precision methods for solving the k-means clustering problems was supplemented by new efficient algorithms.

6. Acknowledgements

Results were obtained in the framework of the state task No. 2.5527.2017/8.9 of the Ministry of Education and Science of the Russian Federation.

REFERENCES

1. S. Vempala and G. Wang: *A spectral algorithm for learning mixtures of distributions*. FOCS. (2002), 841-860.
2. L. Kazakovtsev and A. Antamoshkin: *Genetic Algorithm with Fast Greedy Heuristic for Clustering and Location Problems*. Informatica, **38 (3)** (2014), 229-240.
3. V. I. Orlov, D. V. Stashkov, L. A. Kazakovtsev, I. P. Rozhnov, O. B. Kazakovtseva and I. R. Nasyrov: *Improved method of forming production batchs of electronic components with special quality requirements*. Modern high technology. **1** (2018), 37-42.
4. H. Steinhaus: *Sur la division des corps materiels en parties*. Bull. Acad. Polon. Sci. Cl. III. **IV** (1956), 801-804.
5. S. P. Lloyd: *Least Squares Quantization in PCM*. IEEE Transactions on Information Theory. **28** (1982), 129-137.
6. J. B. MacQueen: *Some Methods of Classification and Analysis of Multivariate Observations*. Proceedings of the 5th Berkley Symposium on Mathematical Statistics and Probability, **1** (1967), 281–297.
7. V. I. Orlov, L. A. Kazakovtsev, I. P. Rozhnov, N. A. Popov and V. V. Fedosov: *neighbourhood search algorithm for k-means clustering*. IOP Conf. Series: Materials Science and Engineering. **450** Article ID 022035 (2018), DOI:10.1088/1757-899X/450/2/022035.
8. B. B. Bhusare and S. M. Bansode: *Centroids Initialization for K-Means Clustering using Improved Pillar Algorithm*. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). **3 Issue 4** (2014).
9. S. Wang: *An Improved K-means Clustering Algorithm Based on Dissimilarity*. International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), Shenyang, China IEEE, (2013).
10. S. Mahmud, M. Rahman and N. Akhtar: *Improvement of K-means clustering algorithm with better initial centroids based on weighted averagen*. 7th International Conference on Electrical and Computer Engineering. IEEE. (2012), ISBN 9781467314367. DOI:10.1109/icece.2012.6471633.

11. K. A. Abdul Nazeer and M. P. Sebastian: *Improving the Accuracy and Efficiency of the k-means Clustering Algorithm*. Proceedings of the World Congress on Engineering **I** WCE 2009, July 1 - 3, 2009, London, U.K.,(2009).
12. P. Hansen and N. Mladenovic: *J-Means: a new local search heuristic for minimum sum of squares clustering*. Pattern Recognition. **34 Issue. 2**,(2001) 405-413 DOI:10.1016/s0031-3203(99)00216-2.
13. A. Dempster, N. Laird and D. Rubin: *Maximum likelihood estimation from incomplete data*. Journal of the Royal Statistical Society, Series B. **39** (1977), 1–38.
14. D. V. Stashkov, V. I. Orlov, I. R. Nasyrov and L. A. Kazakovtsev: *Application of the EM algorithm with a spherical Gaussian distribution to the problem of industrial product classification*. Economics and Management Management Systems. **23 (1.1)** (2017), 185–193.
15. D. V. Stashkov, M. N. Gudyma, L. A. Kazakovtsev, I. P. Rozhnov and V. I. Orlov: *Algorithm for Series of Mixture Distribution Separation Problems*. Reshetnevskie chteniya Krasnoyarsk, **1 (21)** (2017), 327–328. (In Russ.).
16. P. Hansen and N. Mladenovic: *Variable Neighborhood Search*. Search Methodology /E.K.Bruke, G.Kendall [eds.]. Springer US. P. (2005), 211–238 DOI:10.1007/0-387-28356-0-8.
17. L. A. Kazakovtsev, A. A. Stupina and V. I. Orlov: *Genetic algorithm modification with greedy heuristics for continuous allocation and classification problems*. Management systems and information technology **2(56)** (2014), 35–39.
18. R. Farahani, M. Hekmatfar and (eds.): *Facility location: Concepts, models, algorithms and case studies*. Berlin Heidelberg:Springer-Verlag. (2009), 429.
19. L. A. Kazakovtsev: *The greedy heuristics method for systems of automatic grouping of objects*. Diss ... Dr. tech. of science. Krasnoyarsk, (2016), 429.
20. L. A. Kazakovtsev, A. N. Antamoshkin and I. S. Masich : *Fast Deterministic Algorithm for EEE Components Classification*. IOP Conf. Series: Materials Science and Engineering. **94** (2015), article ID 012015, 10 P. DOI: 10.1088/1757-899X/04/1012015.
21. P. Hansen, J. Brimberg, D. Urosevic and N. Mladenovic: *Solving large p-median clustering problems by primal dual variable neighborhood search*. Data Mining and Knowledge Discovery. **19 (3)** (2009), 351-375.
22. P. Hansen and N. Mladenovic. *Neighborhood Search*. Search Methodology /E.K. Bruke, G. Kendall [eds.]. Springer US. (2005), 211-238 DOI:10.1007/0-387-28356-0-8.
23. P. Hansen and N. Mladenovic : *Variable neighborhood search: principles and applications* Eur. J. Oper. Res, **130** (2001), 449-467.
24. Y. Kochetov, N. Mladenovic and P. Hansen: *Local search with alternating surroundings*. Discrete. analysis and research operas. **s. 2, 10:1** (2003), 11-43.
25. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml], access date 28.03.2019.
26. Clustering basic benchmark [http://cs.joensuu.fi/sipu/datasets], access date 28.03.2019.
27. W. Sheng, and X. Liu: *A genetic k-medoids clustering algorithm*. Journal of Heuristics. **12**. 6. (2006). 447-466.

Ivan P. Rozhnov
Siberian State University of Science and Technology
Department of Systems Analysis and Operations Research
prosp. Krasnoyarskiy Rabochiy, 31
660039 Krasnoyarsk, Russia
ris2005@mail.ru

Victor I. Orlov
Siberian State University of Science and Technology
Department of Systems Analysis and Operations Research
prosp. Krasnoyarskiy Rabochiy, 31
660039 Krasnoyarsk, Russia
ttc@krasmail.ru

Lev A. Kazakovtsev
Siberian State University of Science and Technology
Department of Systems Analysis and Operations Research
prosp. Krasnoyarskiy Rabochiy, 31
660039 Krasnoyarsk, Russia
levk@bk.ru