Application of Algorithms with Variable Greedy Heuristics for k-Medoids Problems

Lev Kazakovtsev and Ivan Rozhnov
Reshetnev Siberian State University of Science and Technology
prosp. Krasnoyarskiy Rabochiy 31, Krasnoyarsk 660031, Russia
Siberian Federal University, prosp.Svobodny 79, Krasnoyarsk 660041, Russia
E-mail: levk@bk.ru

Keywords: clustering algorithms, VNS, k-medoids, greedy heuristic method

Received: March 28, 2019

Progress in location theory methods and clustering algorithms is mainly targeted at improving the performance of the algorithms. The most popular clustering models are based on solving the p-median and similar location problems (k-means, k-medoids). In such problems, the algorithm must find several points called cluster centers, centroids, medoids, depending on the specific problem which minimize some function of distances from known objects to the centers. In the the k-medoids problem, the centers (medoids) of the cluster must coincide with one of the clustered objects. The problem is NP-hard, and the efforts of researchers are focused on the development of compromise heuristic algorithms that provide a fairly quick solution with minimal error. In this paper, we propose new algorithms of the Greedy Heuristic Method which use the idea of the Variable Neighborhood Search (VNS) algorithms for solving the k-medoids problem (which is also called the discrete p-median problem). In addition to the known PAM (Partition Around Medoids) algorithm, neighborhoods of a known solution are formed by applying greedy agglomerative heuristic procedures. According to the results of computational experiments, the new search algorithms (Greedy PAM-VNS) give more accurate and stable results (lower average value of the objective function and its standard deviation, smaller spread) in comparison with known algorithms on various data sets.

Povzetek: Avtorji predlagajo nove algoritme za reševanje problema lokacije k-medoidov in gručenja.

1 Introduction

The rapid development of artificial intelligence systems using, inter alia, methods of automatic data grouping (clustering) and methods of location theory, as well as increasing requirements for economic efficiency in all branches, creates a request for the creation of new algorithms with higher requirements for accuracy of the result.

The attempts to discover a universal and, at the same time, exact method for solving most popular location and clustering problems (k-means, k-medoids, etc.), which guarantees the global optimum of the objective function, in the case of a large amount of input data has been recognized as unpromising. The efforts of researchers focused on the development of compromise heuristic algorithms that give a quick solution [1]. The heuristic algorithms or procedures, also called “heuristics” in the literature, are algorithms that do not have a rigorous justification, but gives an acceptable solution to the practically important problems. The so-called "greedy" algorithms are also heuristics. On each iteration, the greedy algorithm selects the best solution from a certain neighborhood (subset of intermediate solutions). At the same time, some of the practically important clustering problems require such a solution which is very close to the exact solution of the problem, and also stable during repeated runs of the randomized algorithm, reproducible and, therefore, verifiable. The problems should be solved online within a limited time. Such problems include, for example, the problem of forming special batches of semiconductor devices in the specialized testing centers [1], where the need to obtain stable results is due to the requirement of reproducibility and verifiability of calculation results that are part of the production process involving two parties with different interests: the manufacturer and the test Centre.

The ensemble (collective) approach [2] allows reducing the dependence of the final decision on the selected parameters of the original models and algorithms and obtaining a more stable solution [3], or isolating "controversial" objects for which different clustering models give a contradictory result, into a separate class. The k-medoids problem is a convenient model for building clustering algorithm ensembles due to the adaptability of the model to the use of various measures of the distance between objects.

The overall aim of the continuous location problem [4] is to find the location of one or several points (centers, centroids, medoids) in continuous space. There is an intermediate class of problems that are actually discrete (the number of possible locations of the searched points is finite), operating with concepts characteristic of the continuous problems. In particular, such is the the k-medoids problem [5, 6] (also called the discrete p-median problem [7] in the scientific literature). The main parameters of all such problems are the coordinates of the objects and the distances between them [8-10]. The aim of the continuous p-median problem [8] is to find k points (centers, centroids, medians, cluster medoids), such that the sum of weighted distances from N known points, called demand points, consumers, objects or data vectors depending on the formulation of a specific problem, to the nearest of the k centers reaches its minimum.

The allocation problems with Euclidean, Manhattan (rectangular), Chebyshev metrics are well studied (all these metrics are particular cases of metrics based on Minkowski lp-norms [11]), and many algorithms have been proposed for solving the Weber problem for these metrics. In particular, the well-known Weisfeld procedure [12] was generalized for metrics based on Minkowski norms.

If the distance is Euclidean $L(X_j, A_i) = \sqrt{\sum_{k=1}^{d}(x_{j,k} - a_{i,k})^2}$, we have the p-median problem. Here, $X_j=(x_{j,1},\dots,x_{j,k})$ $\forall j = \overline{1,p}$, $A_i=(a_{i,1},\dots,a_{i,d})$ $\forall i = \overline{1,N}$. If the squared Euclidean metric is used, $L(X_j, A_i) = \sum_{k=1}^{d}(x_{j,k} - a_{i,k})^2$, we have the k-means problem. Vectors $A_1,\dots,A_N$ are data vectors in a d-dimensional space, $A_i=(a_{i,1},\dots,a_{i,d})$, $A_i \in \mathbb{R}^d$ $A_i \in \mathrm{R}^d$.

In the k-medoids model and problem, cluster centers $X_j=(x_{j,1},\dots,x_{j,k})$ called medoids, are searched among the known points A_i, and this is a discrete optimization problem.

The most popular algorithm for the k-medoid problem, Partitioning Around Medoids (PAM) algorithm, was created by L. Kaufman and P. J. Rousseeuw) [13]. It is very similar to the k-means algorithm. Both algorithms divide a lot of objects into groups (clusters) and both are based on attempts to minimize the error (total distance) on each iteration. The PAM algorithm works with medoids, objects that are part of the original set and representing the cluster in which they are included, and the k-means algorithm works with centroids, which are artificially created objects representing a cluster. The PAM algorithm divides a set of N objects into k clusters (k is a parameter of the algorithm). This algorithm operates the pre-calculated distance matrix between objects, its aim is to minimize the distance between the medoid of each cluster and other objects included in the same cluster.

For discrete optimization problems, the local search methods are the most natural and visual [14]. Such problems include the location problems, building networks, schedules, etc. [15-18]. The standard local descent algorithm starts with some initial solution x_0 (in our case, the initial set of medoids) chosen randomly or with the use of some additional algorithm. At each step of a local descent, the current solution is transformed into to the neighboring solution with a smaller value of the objective function until a local optimum is reached. At each step of local descent, the function of neighborhood O defines a set of possible directions of local search. Very often this set consists of several elements and there is a certain freedom in choosing the next solution. On the one hand, when choosing a neighborhood, it is desirable to have a set of $O(X)$ as small as possible in order to reduce the complexity of a single step. On the other hand, a wider neighborhood can lead to a better local optimum. A possible way to resolve this contradiction is to develop complex neighborhoods, the size of which can be varied during local search [19].

In this paper, we propose the use of local search algorithms that contain greedy agglomerative heuristic procedures, as well as the well-known PAM algorithm, using an idea of the Variable Neighborhood Search (VNS) [20]. It is shown that new VNS algorithms have advantages over the standard PAM algorithm and are competitive in comparison with the known genetic algorithms of the Greedy Heuristics Method for the considered problem [21].

2 Idea of new algorithms

Local search methods have been further developed into metaheuristics [22]. We consider one of them, called the Variable Neighborhoods Search [23, 24]. The idea is to systematically vary the neighborhood function during a local search. Flexibility and efficiency explain its competitiveness in solving NP-hard problems, in particular, p-median problems [25], clustering and location problems [26, 27].

Let us denote by N_k, k=1,..k_{max}, the finite set of neighborhood functions preselected for local search. The proposed method with variable neighborhoods relies on the fact that a local minimum in one neighborhood is not necessarily a local minimum in another neighborhood, and the global minimum is the local minimum in all neighborhoods [14]. In addition, on average, local minima are closer to the global than a randomly selected point, and they are located close to each other. This allows us to narrow the search area for a global optimum using information about local optimums already detected. This hypothesis forms the basis for various crossover operators for genetic algorithms [28] and other approaches.

The deterministic local descent with variable neighborhoods (VND) implies a fixed order of changing neighborhoods and finding a local minimum relative to each of them. Probabilistic local descent with variable neighborhoods differs from the previous VND method by a random selection of points from the neighborhood $O_k(X)\in N_k$. The stage of finding the best point in the neighborhood is omitted. The probabilistic algorithms are most productive in solving problems of large dimension, when the use of a deterministic version requires too much machine time to perform one iteration.

The basic local search scheme with variable neighborhoods is a combination of the two previous options [23].

VNS algorithm

```
Step 1. Choose the neighborhoods $O_k$, $k = 1, .. k_{max}$, and the starting point $x$.
Step 2. Repeat until the stopping criterion is satisfied.
    2.1. $k \leftarrow 1$.
    2.2. Repeat until $k \leq k_{max}$:
        2.2.1. Randomly select a point $x' \in O_k(x)$;
        2.2.2. Apply a local descent from the starting point $x'$. The resulting local optimum is denoted by $x''$;
        2.2.3. if $F(x'') < F(x)$, then it is assumed that $x \leftarrow x''$, $k \leftarrow 1$, otherwise $k \leftarrow k + 1$.
```

We can use the time limitation or maximum number of iterations as the stop criterion. In the case of large-scale problems, the complexity of performing one iteration becomes very large and new approaches are needed to develop effective local search methods.

A popular idea in solving continuous clustering problems is the use of genetic algorithms (GA) and other evolutionary approaches to improve the results of local search [29-31]. Many of these evolutionary algorithms recombine the initial solution obtained by one of the simple local search algorithms.

The PAM procedure consists of two phases: BUILD and SWAP:

- In the BUILD phase, primary clustering is performed, during which k objects are successively selected as medoids.

- The SWAP phase is an iterative process in which the algorithm makes attempts to improve some of the medoids. At each iteration of the algorithm, a pair is selected (medoid and non-medoid) such that replacing the medoid with a non-medoid object gives the best value of the objective function (the sum of the distances from each object to the nearest medoid). The procedure for changing the set of medoids is repeated as long as there is a possibility of improving the value of the objective function.

Algorithm 1 (PAM procedure).

```
Initialization (if needed):
1. Select $k$ objects as a set of medoids (if such set is not given).
2. Build a distance matrix if needed.
*Build Phase:*
3. Assign each object to the nearest medoid.
*Swap Phase:*
4. For each cluster, find objects that reduce the average distance, and if there are such objects, select those that reduce it most strongly, as a medoid.
5. If at least one medoid has changed, return to Step 3, otherwise stop the algorithm.
```

The following algorithm is the basic algorithm of the Greedy Heuristics Method [1] for clustering problems. His idea is that initially some unacceptable solution with an excessive number of centers / medoids is selected, which is then gradually reduced to a solution with a given number of centers.

Algorithm 2. Basic greedy agglomerative heuristic procedure.

```
Required: the initial number of clusters $K$, the required number of clusters $k < K$.
1. Randomly choose an initial solution with $K$ cluster centers $S = \{X_1, ..., X_k\}$ if it is not given.
2. Execute Algorithm 1 with the initial solution $S$, store a new (improved) solution to $S$.
3. If $K = k$, then stop.
4. For each $i' \in \{\overline{1,K}\}$ perform:
    4.1. Get a truncated set $S \leftarrow S \setminus \{X_{i'}\}$.
    4.2. Run Algorithm 1 with the initial solution $S'$. In this case, Algorithm 1 is limited to only one iteration. Store the achieved value of the objective function (1) to $F'_{i'}$.
    4.3. Next iteration of loop 4.
5. Find the index $i'' = \arg\max_{i'=\overline{1,k}} F_{i'}$.
6. Get a truncated set $S \leftarrow S \setminus \{X_{i''}\}$, improve it with Algorithm 1, then go to step 3.
```

This procedure forms the basis for three new procedures that use the centers/medoids of some second known feasible solution to compile an intermediate infeasible solution with an excessive number of clusters K.

Algorithm 3. Greedy procedure # 1.

```
Given: multiple cluster centers $S' = \{X'_1, ..., X'_k\}$ and $S'' = \{X''_1, \ldots, X''_k\}$.
1. For each $i' \in \{\overline{1,k}\}$ perform:
    1.1. Combine element by element sets $S'$ and $S''$: $S \leftarrow S' \cup \{X''_{i'}\}$.
    1.2. Run the basic greedy heuristic (Algorithm 2) with $S$ as the initial solution. The result obtained (the resulting set, as well as the value of the objective function) is memorized.
2. Return the best (by the value of the objective function) solution among the solutions obtained in step 1.2.
```

A simpler, but more demanding in terms of computing resources version of the similar algorithm is presented below.

Algorithm 4. The greedy procedure # 2.

```
1. Form a unified set $S \leftarrow S' \cup S''$.
2. Run Algorithm 2 with $S$ as the initial solution.
```

An intermediate variant is suggested in [21]. There, the Algorithm 2 starts from the set S' united with a randomly chosen subset of S.

Algorithm 5. The greedy procedure # 3.

```
1. Choose random $r' \in [0,1)$. Assign $r \leftarrow [(k/2-2)\, r'2]+2$. Here, [.] is the integer part.
2. Repeat $k$-$r$ times:
    2.1. Form a randomly selected subset of $S'''$ of the $r$ elements of the set $S''$. Join the sets $S \leftarrow S' \cup S'''$.
```

2.2. Run Algorithm 2 with this set *S* as the initial solution.

3. Return the best (in terms of the objective function) among the solutions obtained in step 2.2.

These heuristic procedures, which are local search algorithms in the neighborhood of a known ("parent") solution represented by the set *S'*, can be used as a part of various global search strategies. At the same time, as the neighborhoods of this solution *S'* are formed by adding elements from the other known solution *S''* and eliminating the excessive elements from the unified solution using the basic greedy agglomerative heuristic procedure.

The search in such neighborhoods is made by Algorithms 3-5. Thus, these algorithms search in some neighborhoods of the solution *S'*, and the second known solution *S''* is a randomly selected parameter of this neighborhood. The general idea of the new Variable Neighborhood Search algorithms neighborhoods for solving the k-medoids problem is given below:

```
Algorithms 6. PAM-VNS.
  1. Run Algorithm 1 from a random initial solution, store the resulting set of medoids to S.
  2. Assign $O \leftarrow O_{start}$ //comment: $O_{start}$ is the initial number of neighborhood type).
  3. Assign $i \leftarrow 0$, $j \leftarrow 0$; (the number of unsuccessful iterations in a particular neighborhood and as a whole by the algorithm).
  4. Run Algorithm 1 from the random initial solution, get the solution S'.
  5. Depending on the value of O (values 1, 2 or 3 are allowed), run Algorithm 3, 4 or 5 with the initial solutions S and S'.
  6. If the result (by the objective function value) is better than S, then replace S with this new result, assign $i=0$, $j=0$, go to Step 5.
  7. Assign $i \leftarrow i+1$;
  8. If $i < i_{max}$, then go to Step 4.
  9. Assign $i \leftarrow 0$, $j \leftarrow j+1$. Switch to a new neighborhood type: $O=O+1$; if $O>3$, then assign $O=1$;
  10. If $j > j_{max}$, or other stop conditions are satisfied (maximum running time), then STOP. Otherwise, go to Step 5.
```

The values of the two control parameters are important: the number of ineffectual searches in the current neighborhood i_{max}, and the number of ineffectual switching of the neighborhoods j_{max}. We used the values $i_{max} = 2k$, $j_{max} = 2$.

In addition, important control parameter O_{start}, which specifies the number of the starting neighborhood type. We performed our experiments with all its possible values (1, 2 and 3). Depending on this value, the algorithms are designated below, respectively, PAM-VNS1, PAM-VNS2, PAM-VNS3. In these versions of Algorithm 6, the number of elements in *S'* is equal to the number of elements in *S*: $|S|=|S'|=k$. In special versions called PAM-VNS1-R, PAM-VNS2-R, PAM-VNS3-R, the number of elements (medoids) in *S'* is chosen randomly, $S' \in \{\overline{2,2k}\}$.

3 Computational experiments

In the description of the computation experiments, we used the following abbreviations of the algorithm names: PAM is the classical PAM algorithm in multi-start mode; PAM-VNS1, PAM-VNS2, PAM-VNS3, PAM-VNS1-R, PAM-VNS2-R, and PAM-VNS3-R are variations of Algorithm 6; GA-FULL is the genetic algorithm with a greedy heuristic for the k-medoids problem [1]; GA-ONE is a new genetic algorithm with greedy heuristic [1] where Algorithm 3 is used as a crossing-over procedure.

As test data sets for our experiments, we used the results of non-destructive test tests of prefabricated production batches of semiconductor devices (Tables 1-7), datasets from repositories UCI [32] and Clustering basic benchmark [33]. In our experiments, we used the DEXP computing system (4-core Intel® Core ™ i5-7400 CPU 3.00 GHz, 8 GB of RAM).

For all data sets, 30 attempts were made with each of the 9 algorithms. In every attempt, we fixed the best achieved results. The best values of the objective function (minimum value, mean value, median value and standard deviation) are highlighted in bold italics, the smallest of the best values is additionally highlighted. We used the T-test and the Wilcoxon signed rank test [34, 35] (significance level 0.01 for both tests). Note: "↑", "⇑": the advantage of the best of new algorithms over known algorithms is statistically significant ("↑" for the t-test, and "⇑" for the Wilcoxon test); "↓", "⇓": the disadvantage of the best new algorithms compared to known algorithms is statistically significant; "↕", "⇕": advantage or disadvantage is statistically insignificant.

Table 4 presents the results of the new algorithm in comparison with known evolutionary algorithms that have worked well in solving this problem [1].

In Table 4, we use the following abbreviations [1]:

- GA a genetic algorithm with uniform stochastic crossingover procedure,
- GAGH is the genetic algorithm with greedy heuristic #3 as crossingover procedure,
- LS is the local search by PAM algorithm in

Algorithm	Objective function value (sum of distances)			
	Min (the best attempt)	Average among 30 attempts	Median	Standard deviation
PAM	1 654,4	1 677,4	1679,5	12,2445
PAM-VNS1	***1 554,3***	***1 566,7***	***1565,7***	7,4928
PAM-VNS2	1 558,0	***1 566,1***	***1566,5***	***5,0686***
PAM-VNS3 ↑⇑	***1 555,1***	***1 563,9***	***1564,9***	***3,9161***
GA-FULL	1 599,2	1 637,6	1636,2	25,5365
GA-ONE	1 589,9	1 614,8	1615,4	13,5342

Table 1: Comparative results of computational experiments with data set 3OT122A (767 data vectors, 13 attributes) 10 clusters, 60 seconds for each attempt, 30 attempts, Manhattan distance.

Algorithm	Objective function value (sum of distances)			
	Min (the best attempt)	Average among 30 attempts	Median	Standard deviation
PAM	50 184,0	50 883,7	50 693,0	472,441
PAM-VNS1 ↑⇑	***45 440,4***	***45 553,0***	***45 496,6***	***95,800***
PAM-VNS2	45 453,7	***45 657,7***	***45 648,4***	153,329
PAM-VNS3	***45 444,4***	***45 637,9***	***45 594,4***	177,586
GA-FULL	46 660,9	48 391,2	48 341,9	845,084
GA-ONE	47 081,3	48 125,9	47 965,0	766,566

Table 2: Comparative results of computational experiments with data set 5514BC1T2-9A5 (91 data vectors, 173 attributes) 10 clusters, 60 seconds for each attempt, 30 attempts, Manhattan distance.

Algorithm	Objective function value (sum of distances)			
	Min (the best attempt)	Average among 30 attempts	Median	Standard deviation
PAM	50 184,0	50 883,7	50 693,0	472,441
PAM-VNS1 ↑⇑	***45 440,4***	***45 553,0***	***45 496,6***	***95,800***
PAM-VNS2	45 453,7	***45 657,7***	***45 648,4***	153,329
PAM-VNS3	***45 444,4***	***45 637,9***	***45 594,4***	177,586
GA-FULL	46 660,9	48 391,2	48 341,9	845,084
GA-ONE	47 081,3	48 125,9	47 965,0	766,566

Table 3: Comparative results of computational experiments with data set 1526TL1 (1234 data vectors, 157 attributes) 10 clusters, 60 seconds for each attempt, 30 attempts, Manhattan distance.

multistart mode,

- GA FIX is the genetic algorithm with recombination of fixed-length subsets [36],
- Determ.GH is the deterministic algorithm with greedy heuristic [1] built on the principles of the Information Bottleneck Clustering.
- For some of the datasets, we performed our computational experiments with various number of clusters and various distance metrics (Tables 5-7).

For the genetic algorithms, we used the population size NPOP starting from NPOP=20. In [21], authors show that smaller populations (NPOP<10) in the genetic algorithms with the greedy agglomerative crossingover procedure decrease the accuracy of the result, and larger populations (NPOP>50) slow down the algorithm which also decreases the accuracy.

In all genetic algorithms, we used the simple tournament selection. Traditionally [30], such algorithms do not contain any mutation operator.

Algorithm	Objective function value (sum of distances)			
	Min (the best attempt)	Average among 30 attempts	Median	Standard deviation
PAM	64 232,0	66 520,2	66 776,2	991,994
PAM-VNS1 ↕⇕	***55 361,8***	***55 363,9***	56 004,1	***2,457***
PAM-VNS2	***55 361,8***	55 858,4	55 904,3	359,416
PAM-VNS3	55 383,8	55 755,0	***55 662,5***	353,947
GA-FULL	58 789,3	60 629,5	61 069,2	1187,09
GA-ONE	58 300,2	60 165,4	59 689,1	1388,62
GAGH+LS	***55 361,8***	55 364,1	55 754,2	6,2204
GAGH	***55 361,8***	***55 361,8***	***55 622,3***	***7,8E-12***
GA FIX	***55 361,8***	55 452,7	55 814,1	240,563
GA classical	***55 361,8***	55 364,1	***55 638,9***	6,220
Determ GH	55 998,2	55 998,2	56 199,4	***0,000***

Table 4: Comparative results of computational experiments with data set 1526TL1 (1234 data vectors, 157 attributes) 10 clusters, 60 seconds for each attempt, 30 attempts, squared Euclidean distance.

Algorithm	Objective function value (sum of distances)			
	Min (the best attempt)	Average among 30 attempts	Median	Standard deviation
PAM	2 688,57	2 704,17	2 702,58	12,3308
PAM-VNS1 ↑⇑	***2 607,21***	***2 607,25***	***2 607,21***	***0,1497***
PAM-VNS2	***2 607,21***	***2 607,43***	***2 607,21***	***0,4303***
PAM-VNS3	***2 607,21***	***2 607,34***	***2 607,21***	***0,4159***
GA-FULL	***2 608,22***	2 624,97	2 625,77	9,5896
GA-ONE	***2 608,69***	2 625,18	2 624,57	10,7757

Table 5: Comparative results of computational experiments with data set Ionosphere (351 data vectors, 35 attributes) 10 clusters, 60 seconds for each attempt, 30 attempts, Manhattan distance.

Algorithm	Objective function value (sum of distances)			
	Min (the best attempt)	Average among 30 attempts	Median	Standard deviation
PAM	319,84	***343,44***	***346,23***	***15,300***
PAM-VNS1	278,63	390,43	367,03	82,609
PAM-VNS2	333,26	471,15	450,34	100,259
PAM-VNS3	***273,91***	354,98	***352,75***	54,244
PAM-VNS1-R	301,91	428,14	398,28	129,216
PAM-VNS2-R	384,62	475,92	470,64	53,097
PAM-VNS3-R ↑⇑	***265,96***	***325,49***	***317,94***	42,144
GA-FULL	315,57	383,41	365,04	60,149
GA-ONE	343,21	433,01	424,73	66,036

Table 6: Comparative results of computational experiments with data set Mopsi-Joensuu (6015 data vectors, 2 attributes) 20 clusters, 60 seconds for each attempt, 30 attempts, Euclidean distance.

4 Conclusion

The results of our computational experiments showed that the new search algorithms in alternating neighborhoods (PAM-VNS) can outperform known algorithms and give more stable results (a lower median and average values and / or standard deviation of the objective function, a smaller spread of the achieved values) and, consequently, better performance in comparison with known algorithms. The comparative efficiency of the new algorithm and its modifications on several data sets has been experimentally proven.

Algorithm	Objective function value (sum of distances)			
	Min (the best attempt)	Average among 30 attempts	Median	Standard deviation
PAM	10 763,0	10 822,4	10 833,0	***47,127***
PAM-VNS1	10 357,0	***10 530,9***	***10 563,5***	122,962
PAM-VNS2	10 803,0	11 107,1	11 079,5	174,118
PAM-VNS3	10 429,0	10 594,6	***10 575,0***	114,719
PAM-VNS1-R	10 400,0	10 659,0	10 664,5	161,298
PAM-VNS2-R	10 891,0	11 097,0	11 096,5	187,911
PAM-VNS3-R ↕⇕	***10 310,0***	10 623,3	***10 597,5***	214,129
GA-FULL	***10 252,0***	***10 381,3***	***10 393,0***	***72,911***
GA-ONE	10 944,0	11 098,0	11 064,5	112,081

Table 7: Comparative results of computational experiments with data set Chess (3196 data vectors, 37 binary attributes) 50 clusters, 60 seconds for each attempt, 30 attempts, squared Euclidean distance.

However, for some of the considered datasets, the genetic algorithms show their advantages over new algorithms. Genetic algorithms show the best results in the case of a complex relief of the objective function, the presence of plateaus, due to diversity in the population, as revealed in some of the cases considered. One of the most important shortcomings of such algorithms is the possible “dumping” of the entire population into the region of attraction of a single local minimum. Probably, new VNS-algorithms allow us to avoid this drawback due to the continuous random generation of new solutions, which are parameters of the search neighborhood, and allow us to "jump out" of the attraction region.

Therefore, the interest for further research is the combined approach, combining the presence of a certain population with the possibility of generating random solutions, which play a role similar to that of the mutation operator in traditional genetic algorithms.

Acknowledgement

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] Kazakovtsev L.A. (2016) *The greedy heuristics method for systems of automatic grouping of objects*. Diss ... Dr. tech. of science. Krasnoyarsk. Siberian Federal University.

[2] Ghosh J., Acharya A. (2011) Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 1(4), pp..305−315. https://doi.org/10.1002/widm.32.

[3] Rozhnov I., Orlov V., Kazakovtsev L. (2018) Ensembles of clustering algorithms for problem of detection of homogeneous production batches of semiconductor devices. *CEUR-WS,* Vol.2098, pp.338-348. https:// http://ceur-ws.org/Vol-2098/paper29.pdf

[4] Drezner Z., Hamacher H. (2004) *Facility location: applications and theory*. Berlin:Springer-Verlag.

[5] Struyf A., Hubert M., Rousseeuw P. (1997) Clustering in an Object-Oriented Environment. *Journal of Statistical Software*, Issue 1 (4), pp.1-30. https://doi.org/ 10.18637/jss.v001.i04.

[6] Kaufman L., Rousseeuw P.J. (1990) Finding groups in data: an introduction to cluster analysis. New York:Wiley. https://doi.org/10.1002/9780470316801

[7] Moreno-Perez J.A., Roda Garcia J.L., Moreno-Vega J.M. (1994) *A Parallel Genetic Algorithm for the Discrete p-Median Problem*. Studies in Location Analysis, Issue 7, pp.131-141.

[8] Wesolowsky, G. (1993) The Weber problem: History and perspectives. *Location Science*, No.1, pp.5-23.

[9] Drezner Z., Wesolowsky G.O. (1978) A Trajectory Method for the Optimization of the Multifacility Location Problem with lp Distances. *Management Science*, Vol.24, pp.1507-1514. https://doi.org/10.1287/mnsc.24.14.1507

[10] Farahani R., Hekmatfar M. (2009). *Facility location: Concepts, models, algorithms and case studies*. Berlin Heidelberg:Springer-Verlag. https://doi.org/10.1080/13658816.2010.528422

[11] Deza M.M., Deza E. (2013) Metrics on Normed Structures. *Encyclopedia of Distances*. Berlin Heidelberg: Springer, pp.89-99. https://doi.org/10.1007/978-3-642-30958-85.

[12] Weiszfeld, E. (1937) Sur le point sur lequel la somme des distances de n points donnes est minimum. *Tohoku Mathematical Journal*, Vol.43, No.1, pp.335-0386. https://link.springer.com/article/10.1007/s10479-008-0352-z

[13] Kaufman L. and Rousseeuw P.J. (1987) Clustering by means of Medoids. *Statistical Data Analysis Based on the L1-Norm and Related Methods*, Springer US. pp. 405-416.

[14] Kochetov Yu., Mladenovic N., Hansen P. (2003) Local search with alternating neighborhoods. *Discrete analysis and operations research*, Series 2, Vol.10(1), pp.11-43.

[15] Nicholson T. A. J. (1965) A sequential method for discrete optimization problems and its application to the assignment, traveling salesman and tree scheduling problems. *J. Inst. Math. Appl*, Vol.13, pp.362-375. https://doi.org/10.1093/imamat/3.4.362

[16] Page E. S. (1965) On Monte Carlo methods in congestion problems. I: Searching for an optimum in discrete situations. *Oper. Res.* Vol.13(2), pp. 291-299. https://doi.org/10.1287/opre.13.2.291

[17] Kernighan B. W., Lin S. (1970) An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.* Vol.49. pp.291-307. https://doi.org/10.1002/j.1538-7305.1970.tb01770.x

[18] Rastrigin L.A. (1978) Random search - specificity, stages of history and prejudice. *Questions of*

cybernetics. M.: Nauch. Council on the complex problem "Cybernetics" of the USSR Academy of Sciences, Vol. 33, pp.3-16.

[19] Kochetov Yu.A. (2010) Local search methods for discrete placement problems. *Dis. Doctors of Physical and Mathematical Sciences*. Novosibirsk.

[20] Hansen P., Mladenovic N., Bruke E.K., Kendall G. (2014) *Variable Neighborhood Search. Search Methodology*. Springer US. P.211-238. https://doi.org/10.1007/0-387-28356-0_8.

[21] Kazakovtsev, L.A., Antamoshkin, A.N. (2014) Genetic Algorithm with Fast Greedy Heuristic for Clustering and Location Problems. *Informatica*, Vol.38(3), pp.229-240.

[22] Osman I. H., Laporte G. (1996) Metaheuristics: a bibliography. *Ann. Oper. Res*, Vol.63. pp.513-628. https://doi.org/10.1007/BF02125421.

[23] Mladenovic N., Hansen P. Variable neighborhood search. *Comput. Oper. Res*, Vol.24, P.1097-1100. https://doi.org/10.1016/S0305-0548(97)00031-2.

[24] Hansen P., Mladenovic N. (2001) Variable neighborhood search: principles and applications (invited review). *European J. Oper. Res*, Vol.130(3), pp.449-467. https://doi.org/10.1016/S0377-2217(00)00100-4

[25] Garcia-Lopez F., Melian-Batista B., Moreno-Perez J.A., Moreno-Vega M. (2002) The parallel variable neighborhood search for the p-median problem. *Journal of Heuristics*, Vol.8, pp. 375-388 (2002). https://doi.org/10.1023/A:1015013919497.

[26] Brimberg J., Mladenovic N. (1996) A variable neighborhood algorithm for solving the continuous location-allocation problem. *Stud. Locat. Anal.* V. 10. pp. 1-12.

[27] Hansen P., Mladenovic N., Perez-Brito D. (2001) Variable neighborhood decomposition search. *J. Heuristics*, Vol.7 (4), pp. 335-350. https://doi.org/ 10.1023/A:1011336210885.

[28] Goldberg D. E. (1989) *Genetic algorithms in search, optimization, and machine learning. Reading*, MA: Addison-Wesley. https://doi.org/10.5555/534133.

[29] Houck C.R., Joines J. A., G.Kay. M. (1996) Comparison of Genetic Algorithms, Random Restart, and Two-Opt Switching for Solving Large Location-Allocation Problems. *Computers and Operations Research*, Vol. 23, pp. 587-596. https://doi.org/10.1016/0305-0548(95)00063-1.

[30] Alp O., Erkut E., Drezner Z. (2003) An Efficient Genetic Algorithm for the p-Median Problem. *Annals of Operations Research.* Vol.122, pp.21-42, https://doi.org/10.1023/A:1026130003508. (2003).

[31] Neema M.N., Maniruzzaman K.M., Ohgai A. (2011) New Genetic Algorithms Based Approaches to Continuous p-Median Problem. *Netw. Spat. Econ.*, Vol.11, pp.83-99, https://doi.org/10.1007/s11067-008-9084-5.

[32] UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. access date 28.03.2019.

[33] Clustering basic benchmark [http://cs.joensuu.fi/ sipu/datasets], access date 28.03.2019.

[34] Wilcoxon F. (1945) Individual comparisons by ranking methods. *Biometrics Bulletin*, Vol.1(6), pp. 80–83. https://doi.org/10.2307/3001968.

[35] Derrac J., Garcia S., Molina D., Herrera F. (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, Vol. 1(1), pp. 3-18, https://doi.org/j.swevo.2011.02.002.

[36] Sheng W., Liu X. (2006) A genetic k-medoids clustering algorithm. *Journal of Heuristics*. Vol.12, No.6. P. 447-466. https://doi.org/10.1007/s10732-006-7284-z.