FAST DETERMINISTIC ALGORITHM FOR EEE COMPONENTS CLASSIFICATION*

L A Kazakovtsev[1], A N Antamoshkin[1,2] and I S Masich[1]
[1] Department of System Analysis and Operation Research, Siberian State Aerospace University named after academician M.F. Reshetnev, Krasnoyarsk, Russia
[2] Department of Economics and Information Technology of Management, Siberian Federal University, Krasnoyarsk, Russia

E-mail: levk@bk.ru

Abstract. Authors consider the problem of automatic classification of the electronic, electrical and electromechanical (EEE) components based on results of the test control. Electronic components of the same type used in a high-quality unit must be produced as a single production batch from a single batch of the raw materials. Data of the test control are used for splitting a shipped lot of the components into several classes representing the production batches. Methods such as k-means++ clustering or evolutionary algorithms combine local search and random search heuristics. The proposed fast algorithm returns a unique result for each data set. The result is comparatively precise. If the data processing is performed by the customer of the EEE components, this feature of the algorithm allows easy checking of the results by a producer or supplier.

Introduction

Supplying the electronic units of the complex technical systems with EEE components of the proper quality is one of the most important problems for increasing the whole system reliability. Moreover, for reaching the highest reliability of an electronic unit, the EEE components of the same type must have equal characteristics which assure their coherent operation. The highest homogeneity of the characteristics is reached if the EEE components are produced as a single production batch from a single batch of the raw materials. The critically important units are integrated from EEE components manufactured as a special production lots with special quality requirements [1, 2].

The characteristics of a lot of the components are checked via destructive and nondestructive tests [1, 3]. Data of such tests can be used for analyzing the lot homogeneity [3]. For splitting the EEE components into several assumed production batches, the k-means method can be used [4, 5, 6].

American and European EEE component manufacturers produce components of special quality classes, Military and Space [7, 8]. Russian manufacturers do not form a special class of components for use in space systems [1, 2].

The k-means problem can be classified as a continuous problem of the location theory [9, 10, 11]. The aim is to find k points (centers, centroids, medoids) in a d-dimensional space

* Results were obtained in the framework of the state task № 346 of the Ministry of Education and Science of the Russian Federation

such that the total distance from each of the data vectors (known points, measurement result vectors) to the nearest of *k* chosen centers reaches its minimum:

$$F(X_1,...,X_k)=\sum_{i-1}^{N}\min_{x\in\{x_1,...,x_k\}}\|A_i-X\|_2^2. \quad (1)$$

Quality requirements of the EEE components in space systems are so high that the the range of characteristics measured via quality check tests is very narrow and the quality class and assumed production batch of each component in the lot must be determined via analysis of difference (distance) of test result data vectors which slightly exceed the precision of the measurement tools. Thus, results of each measurement form a finite (discrete) set of possible values defined by the measurement tool precision.

The squared Euclidean norm is most popular distance metric used for calculating differences (distances) in a normalized space of characteristics [9]. Using the rectilinear (Manhattan) [12] norm as a distance metric in the k-means model allows to reach results of the same precision as the precision of data vectors. In this case, the value of each coordinate of the result coincides with the value of the same coordinate of one of the data vectors [9, 13]. Moreover, the result of the k-means problem rectilinear metric are more stable under the influence of the outliers in the data which exist due to measurement errors and defective components in the lot. Other approach which allows to achieve the results of the same precision as data vectors is solving the *k*-medoid problem [14, 15]. In this problem, the cluster centers which minimize the total distance are searched among the data vectors only.

The k-means method uses the ALA procedure (Alternating Location-Allocation) which includes two simple steps:

Algorithm 1. ALA procedure.

Required: data vectors $A_1 \ldots A_N$, *k* initial cluster centers $X_1 \ldots X_k$.

1. For each center X_i, determine its cluster C_i as a subset of the data vectors for which this center X_i is the closest one.

2. For each cluster C_i, recalculate its center X_i.

$$X_i = \arg\min_{X} \sum_{y\in C_i}\|x-y\|.$$

3. Repeat Step 1 unless Steps 1, 2 made no change in data.

Traditional usage of the k-means method with squared Euclidean metric (l_2) has one important advantage: in this case, calculating a center of a cluster is a simple problem solved in one iteration as calculating the mean value for each coordinate of all data vectors in the cluster. These mean values are coordinates of the new cluster center [9]. If the *i*th cluster center $X_i=(x_{i,1},\ldots,x_{i,d})$ is a vector in a *d*-dimensional space and data vectors $A_j=(a_{i,1},\ldots,a_{j,d})$, $j=\overline{1,N}$ also have *d* dimensions then the new cluster center is calculated as [9]:

$$x'_{i,k}=\sum\nolimits_{y\in C_i} y_k / |C_i|, k=\overline{1,d}\ .$$

In the ALA procedure with the rectilinear metric, each coordinate of the cluster center is calculated as the median value of this coordinate among all data vectors which belong to the cluster. This process can be described as follows.

Algorithm 2. Calculating *i*th cluster center (median) in case of the l_1 metric.

1. For each $k=\overline{1,d}$ do:

1.1. Sort vectors $A_i = (a_{j,1},...a_{j,d}) \in C_i$ in acceding order of the kth coordinate, form a sequence $a'_{1,k},...,a'_{|C_i|,k}$. Here, $|C_i|$ is the power of the set (cluster).

1.2. Calculate $m = \left[|C_i| / 2 \right]$. Store $x'_{i,k} = a'_{m,k}$. Here, $[\bullet]$ is the integer part.

1.3. Next iteration 1.

2. Return $X'_i = (x'_{i,1},...,x'_{i,d})$

This algorithm returns a value of new center which has values of each coordinate coinciding with a coordinate of some data vector.

In the case of the *k*-medoid problem, procedure of determining of each cluster center requires the exhaustive search among all data vectors in the cluster. However, many researchers propose faster analogous local search procedures [17, 18, 19] which do not guarantee an exact solution.

Except special cases, the k-means and k-medoids problems are *NP* hard and require global search [20].

The result of the ALA procedure depends on the choice of the initial cluster centers. Known k-means++ algorithm [21] has an advantage in comparison with the chaotic choice of the initial centers and guarantees the statistical preciseness of the result $O(\log(p))$. However, such preciseness is insufficient for many practically important problems. For such cases, researchers propose various recombination techniques for initial center sets [9].

The ALA procedure can be optimized with use of many techniques. For example, sampling procedures [22] solve the k-means problem for the randomly selected subset of the data vectors and use the achieved result as an initial set of centers for solving the original problem. Authors propose various algorithms for streaming data processing [4] applicable for big data analysis and many other methods.

The dependence of the results of the ALA procedure on the initial centers seeding is a serious problem for the reproducibility of the classification algorithm results: depending on the initial centers seeding, different algorithm starts classify the same data vectors as elements of various clusters. For the EEE component production batches classification problem, this means that various EEE component belong to the same or different production batches depending on the initial seeding. Thus, an algorithm for solving k-means problem which returns a stable result is preferred.

The Information Bottleneck Clustering method (IBC) is a deterministic method for solving the cluster analysis and classification problems able to achieve perfect results in many cases.[23]. This algorithm starts from considering each data vector as a separate cluster. Then, clusters are removed one-by-one until the desired clusters quantity remains. Each time, the algorithm eliminates such cluster that its elimination gives the smallest increment of the objective function value. For the k-means problem, this algorithm eliminates the cluster center which gives the smallest total distance from data vectors to the closest remaining centers. Such algorithms are extremely slow [23]. The genetic algorithms with greedy heuristic initially designed for the discrete k-median problem on a network [24] are compromise variants. However, they are not determined algorithms. In [25], author proposes an approach for adaptation of these algorithms for the continuous location problems. The idea of such approach can be described as follows [10].

Algorithm 3. Genetic algorithm with greedy heuristic for k-median problems.

Required: data vectors $A_1 \ldots A_N$, population size N_p.

1. Form (randomly or using the k-means++ procedure) N_p various initial solutions $\chi_1,...\chi_{N_p} \subset \{\overline{1,N}\}$, $|\chi_i|=k \forall i=\overline{1,N_p}$. Each of such solutions is a set of *k* data vectors used as the initial solutions of the ALA procedure. Calculate the fitness function value (total distance) $F_{fitness}(\chi)$ for each of the initial solutions using Algorithm 3 and store the fitness function values to $f_1,...,f_{N_p}$.

2. If the stop conditions are satisfied then STOP. Return solution χ_{i^*}, with the smallest value f_i. For the final solution, th ALA procedure (Algorithm 1) runs.

3. Choose randomly two indexes $k_1,k_2 \in \{\overline{1,N}\}, k_1 \neq k_2$.

4. Form an interim solution $\chi_c=\chi_{k_1} \cup \chi_{k_2}$.

5. If $|\chi_c|>k$ then go to Step 7:

6. Calculate $j^*=arg\min_{j\in\chi_c} F_{fitness}(\chi_c \setminus \{j\})$. Exclude j^* from χ_c : $\chi_c=\chi_c\setminus\{j^*\}$. Go to Step 5.

7. If $\exists i \in \{\overline{1,N_p}\}: \chi_i=\chi_c$ then go to Step 2.

8. Choose index $k_3 \in \{\overline{1,N_p}\}$. First, two indexes $k_4,k_5 \in \{\overline{1,N}\}$ are chosen randomly, then if $f_{k_4}>f_{k_5}$ then $k_3=k_4$ else $k_3=k_5$.

9. Replace χ_{k_3} and the corresponding fitness function value: $\chi_{k_3}=\chi_c$, $f_{k_3}=F_{fitness}(\chi_c)$. Go to Step 2.

Steps 5-6 of this algorithm realize the greedy heuristic, the successive elimination of the centers from an interim solution.

An analogous heuristic was proposed by Kuehn and Hamburger in 1963 [16]. The IBC method is based on the same principle of the successive elimination of the clusters from an interim solution [23]. Both method of Kuehn and Hamburger and the IBC chose an unfeasible solution coinciding with the whole data vectors set as the initial solution.

The fitness function for the k-means problem can be calculated for an initial or interim solution as follows:

Algorithm 4. Calculating fitness function value $F_{fitness}(\chi)$.

Required: solution χ.

1. Run Algorithm 1 with initial centers set $\{A_i|i \in \chi\}$ resulting with centers set $\{X_1,...,X_p\}$.

2. Return $F_{fitness}(\chi)=\sum_{i=1}^{N} w_i \min_{j\in\{\overline{1,k}\}} L(X_j,A_i)$.

This algorithm is a computationally intensive procedure. Other approach [25] is based on the immediate usage of the total distance from data vectors to the closest center in the interim solution as the fitness function value for Algorithm 3 . In this case, Step 1 of Algorithm 4 is omitted. In fact, this approach solves a *k*-medoid problem and adjusts the solution with the ALA procedure at the final iteration of the greedy heuristic. Such approach is much faster. However, it reduces the preciseness.

An advantage of the IBC method is its determinancy. This method does not use any random values and each start of the algorithm results in the same set of cluster centers. The quality checks of the EEE components lots is a process which involves two parts, a manufacturer or supplier and a customer or a specialized testing center. The calculation

performed by the customer must be reproducible. The exact reproduction of the results is impossible when using algorithms which include random search elements. The IBC method slows down the calculation.

In [11], authors propose a modification of the greedy heuristic used in Step 6 of Algorithm 3. This method uses points in the *d*-dimensional space instead of data vector indexes as an alphabet of the genetic algorithm. Such points represent the cluster centers. Authors entitle this modification Algorithm with Floating Point Alphabet. Results of the genetic algorithm with this heuristic are much more precise than results of the modification described in [25] (in [25], authors propose solving a k-medoid problem). At the same time, the iterations in Algorithm 3 with this heuristic are performed much faster than Algorithm 4 as the greedy heuristic. Moreover, decrease of the computational complexity of Step 6 of Algorithm 3 makes solving large-scale problems possible. In [11], authors report results of solving problems with up to 560000 data vectors. Such heuristic can be described as follows.

Algorithm 5. Greedy heuristic for the GA with floating point alphabet (used instead of Steps 4-7 of Algorithm 3).

Required: set of data vectors $V=(A_1,...,A_N) \in \mathrm{R}^d$, number *k* of clusters, two "parent" center sets χ_{k_1} and χ_{k_2}, parameter α.

1. Form interim solution $\chi_c=\chi_{k_1} \cup \chi_{k_2}$. Run ALA algorithm from initial solution χ_c. Store its result to χ_c.

2. If $|\chi_c| = k$ then start ALA procedure from initial solution χ_c, STOP and return χ_c.

2.1. Calculate distances to the closest element of χ_c:

$$d_i = \min_{X \in \chi_c} L(X, A_i) \; \forall i = \overline{1,N}.$$

Form clusters around each center in χ_c. $C_i = \arg\min_{X \in \chi_c} L(X, A_i) \; \forall i = \overline{1,N}.$

Calculate distances from each data vector to the second closest center in χ_c.

$$d_i = \min_{Y \in \chi_c \setminus \{C_i\}} L(Y, A_i) \; \forall i = \overline{1,N}.$$

3. For each center $X \in \chi_c$, calculate $\delta_X = F(\chi_c \setminus \{X\}) = \sum_{i:X \in C_i} (D_i - d_i)$.

4.1. Calculate $n_\delta = \max\{[\alpha(|\chi_c| - k)], 1\}$. Sort values δ_X in acceding order and choose a subset $\chi_{e\lim} = \{X_1,...,X_{n_\delta}\}$ of n_δ data vectors with minimal values δ_X.

4.2. For each $j \in \{\overline{2,|\chi_{e\lim}|}\}$ do: if $\exists q \in \{\overline{1,(j-1)}\}$: $L(X_j, X_q) < L_{min}$ then eliminate X_j from set $\chi_{e\lim}$. Here, $L_{\min} = \min_{X \in \chi_c} \{\max\{L(X, X_j), L(X, X_q)\}\}$.

4.3. Set $\chi_c = \chi_c \setminus \chi_{e\lim}$.

4.4. Reallocate data vectors to the closest centers: $C_i^* = \arg\min_{X \in \chi_c} L(X, A_i) \; \forall i = \overline{1,N}.$

4.5. For each $X \in \chi_c$ if $\exists i \in \{\overline{1,N}\}$: $C_i=X$ and $C_i^* \neq X$ then recalculate center X^* of cluster $C_X^{clust} = \{A_i \mid C_i^* = X, i = \overline{1,N}\}$. Store $\chi_c = (\chi_c \setminus \{X^*\}) \cup \{X\}$.

5. Go to Step 2.

Value α is an important parameter of this heuristic. This value determines the percentage of the superfluous cluster centers eliminated in a single iteration. Authors propose value 0.2. Bigger values make the algorithm run faster and reduce its preciseness. Small values α make it work as Algorithm 4 and eliminate a single center at each iteration. We used α=0.25.

This heuristic combines the greedy heuristic [24, 25, 3] with elements of the modified ALA procedure performed at each iteration and allows to eliminate up to 20-25% superfluous cluster centers until the required quantity of centers remains. Algorithm 4 requires performing $p(k_0\text{-}k)$ runs of the ALA procedure (here, k_0 is the initial centers quantity). Algorithm 5 reduces quantity of iterations down to $O(\log(k_0\text{-}k))$. Moreover, each iteration does not require performint the whole ALA procedure. Instead, its separate optimized elements (location – allocation) are performed.

It can be easily seen that if $k_0=N$ then both Algorithm 5 and IBC method start their initial iterations analogously: number of cluster centers coincides with the number of data vectors. Moreover, if $k_0=N$ then choosing th initial centers is not random: all data vectors are chosen as the initial centers. Thus, the following deterministic algorithm can be proposed.

Algorithm 6. New deterministic greedy heuristic algorithm.

Required: set of data vectors $V=(A_1,...,A_N) \in \mathrm{R}^d$, number k of clusters, parameter α.

1. Set $\chi_c = V$.

2. Run Algorithm 5 starting from Step 2.

Such algorithm was successfully applied for middle-scale problems, up to N=6500 data vectors. The results are gathered in Table 1. We used example problems from the UCI library [26] and problems with real data of EEE components examination. For small-scale problems, the results are shown in comparison with the IBC method, genetic algorithm with greedy heuristic and genetic algorithm with recombination of fixed length subsets. We used various distance metrics. In addition, k-medoid problems [14, 15] were solved. For several problems, the results of new algorithm are insignificantly worse than the results of the IBC. At the ame time, time needed for problem solution reduces many times. In Table 1, all results of new algorithm are shown for $\alpha = 0.25$ and $\alpha = 0.001$. Last value makes the algorithm work as an IBC procedure which eliminates exactly one center at a time. At the same time, the approach to the inclusion of elements of the ALA procedure in this greedy heuristic remains unchanged and the new algorithm with $\alpha = 0.001$ works slower than one with $\alpha = 0.25$ and still much faster than the IBC algorithm which uses Algorithm 4 for fitness function evaluation on each iteration. In addition, Table 1 shows results of thee simplified IBC method which evaluates the fitness function value without running ALA or another local search procedures. In fact. This simplified IBC procedure solves a k-medoid problem and then adjusts the result with ALA procedure. Such algorithm is entitled “IBC w/o local search”. Such algorithm can be constructed after excluding Steps 3 and 4.5 from Algorithm 5 with $\alpha = 0.001$. If we remain $\alpha = 0.25$, i.e..we allow simultaneous eliminating several centers, we have a new deterministic algorithm entitled «IBC w/o local search, $\alpha = 0.25$». This version of the algorithm shows the best time and the worst preciseness.

For all inspected problems except those with the Jaccard metric, new algorithm shows the best results among all considered deterministic algorithms except IBC with local search which exceeds new algorithm by preciseness in several problems. However, new algorithm takes much less time. Results of new algorithm are less precise than the results of evolutionary algorithms. However, except problems with the Jaccard metric, the difference does not exceed

2.3% for problems with real data vectors and 3.8% for problems with Boolean data vectors.

Note that new algorithm has one important feature for solving a problem of automatic classification of the EEE components. Such problem is solved [3] as series of the k-means problems with $k = \overline{k_{min}, k_{max}}$ where k_{min}=1 (a single production batch without clusters assumed) and k_{max} is chosen by a decider equal to some reasonable number. Algorithm 6 can be used for the k-means problem with $k=k_{max}$. Then, starting from Step 2, Algorithm 5 can run again for solving the succeeding problems until $k=k_{min}$. Thus, results for all values $k = \overline{k_{min}, k_{max}}$ can be calculated in a single run.

Table 1. Comparative results of new algorithm.

Data set, quantity of data vectors, dimension, data type	Clusters q-ty *k*, metric, problem type	Algorithm	Result	Time, sec.	Std. devition of 10 runs
Chess (UCI), *N*=3197, *d*=50, Boolean	50, l_1, k-сред-них	Local search multistart	10020.2	35	19.11
		GA with fixed subsets recomb.	9328.4	35	6.066
		GA with float. point alphabet	9290	35	9.148
		IBC	-	-	-
		IBC w/o local search	9796	29.24	
		IBC w/o local search, $\alpha = 0.25$	10057	0.696	Determ.
		New algorithm, $\alpha = 0.25$	9649	0.694	Determ.
		New algorithm, $\alpha = 0.001$	9610	34.52	Determ.
Examination of diodes, *N*=701, *d*=18,real	30, l_1 нор-мирo-ван., k-сред-них	Local search multistart	5451.79	35	4,158
		GA with fixed subsets recomb.	5427.32	35	6,31
		GA with float. Point alphabet	5381.55	35	1,46
		IBC	5390.12	2251.1	Determ.
		IBC w/o local search	5494.05	0.686	Determ.
		IBC w/o local search, $\alpha = 0.25$	5483.89	0.111	Determ.
		New algorithm, $\alpha = 0.25$	5420.67	0.0669	Determ.
		New algorithm, $\alpha = 0.001$	5414.06	0.8962	Determ.
Examination of diodes, *N*=701, *d*=18, real.	30, l_2^2 nor-mali-zed., k-medo-ids	Local search multistart	1950.52	7	5,362
		GA with fixed subsets recomb.	1887.29	7	4,11
		GA with float. Point alphabet	1885.79	7	1,457
		IBC	1902.93	1062.1	Determ.
		IBC w/o local search	1917.14	0.637	Determ.
		IBC w/o local search, $\alpha = 0.25$	1959.97	0.0324	Determ.
		New algorithm, $\alpha = 0.25$	1918.56	0.039	Determ.
		New algorithm, $\alpha = 0.001$	1912.43	0.6696	Determ.
Examination of chip 1526LE2, *N*=620, *d*=120, real	30, l_2^2 nor-mali-zed., k-means	Local search multistart	12873.22	30	50,71
		GA with fixed subsets recomb.	12536.19	30	3,11
		GA with float. Point alphabet	12539.5	30	2,59
		IBC	12556.04	4453.9	Determ.
		IBC w/o local search	12727.13	0.5484	Determ.
		IBC w/o local search, $\alpha = 0.25$	12788.47	0.0088	Determ.

		New algorithm, $\alpha = 0.25$	12834.9	0.1258	Determ.
		New algorithm, $\alpha = 0.001$	12682.32	0.7528	Determ.

Data set, quantity of data vectors, dimension, data type	Clusters q-ty *k*, metric, problem type	Algorithm	Result	Time, sec.	Std. devition of 10 runs
MissAmerica1 (UCI), *N*=6480, *d*=16, integer	100, l_2^2, k-сред.	Local search multistart	717488.7	60	1107.24
		GA with fixed subsets recomb.	698055.6	60	460.62
		GA with float. point alphabet	698054.5	60	406.42
		IBC	-	-	-
		IBC w/o local search	715762.6	159.8	Determ.
		IBC w/o local search, $\alpha = 0.25$	716150.6	1.85	Determ.
		New algorithm, $\alpha = 0.25$	707529.5	2.02	Determ.
		New algorithm, $\alpha = 0.001$	703786.4	199.5	Determ.
PimaIndians Diabetes (UCI), *N*=768, *d*=9, real	30, l_2^2, k-means	Local search multistart	2072.425	36	2.4422
		GA with fixed subsets recomb.	2045.196	36	1.7294
		GA with float. point alphabet	2045.365	36	1.238
		IBC	2043.481	1150.1	Determ.
		IBC w/o local search	2096.06	0.81	Determ.
		IBC w/o local search, $\alpha = 0.25$	2144.771	0.031	Determ.
		New algorithm, $\alpha = 0.25$	2087.46	0.042	Determ.
		New algorithm, $\alpha = 0.001$	2056.35	1.058	Determ.
Breast Cancer (UCI), *N*=699, *d*=10, cathegories	20, Jaccard, k-medoids	Local search multistart	184.1	5	0.8725
		GA with fixed subsets recomb.	172.81	5	0.365
		GA with float. point alphabet	172.62	5	0.0787
		IBC	177.5	370.16	Determ.
		IBC w/o local search	174.4	0.8898	Determ.
		IBC w/o local search, $\alpha = 0.25$	174.6	0.8072	Determ.
		New algorithm, $\alpha = 0.25$	175.2	0.437	Determ.
		New algorithm, $\alpha = 0.001$	175.7	0.7352	Determ.
Zoo (UCI), *N*=101, *d*=10, cathegories	10, метрика Жаккара, k-medoids	Local search multistart	6.4285	1	0.0287
		GA with fixed subsets recomb.	6.4118	1	0
		GA with floating point alphabet	6.4118	1	0
		IBC	6.4706	4.533	Determ.
		IBC w/o local search	6.4706	0.015	Determ.
		IBC w/o local search, $\alpha = 0.25$	6.4706	0.010	Determ.
		New algorithm, $\alpha = 0.25$	6.4706	0.009	Determ.
		New algorithm, $\alpha = 0.001$	6.4706	0.012	Determ.
Ionosphere (UCI), *N*=351, *d*=35, real.	10, L_1, k-means	Local search multistart	2530.40	4	2.117
		GA with fixed subsets recomb.	2526.77	4	0.0257
		GA with float. point alphabet	2527.00	4	0.0082
		IBC	2536.06	76.13	Determ.

		IBC w/o local search	2546.54	0.1124	Determ.
		IBC w/o local search, $\alpha = 0.25$	2572.40	0.0198	Determ.
		New algorithm, $\alpha = 0.25$	2531.03	0.025	Determ.
		New algorithm, $\alpha = 0.001$	2533.31	0.151	Determ.

Conclusion

Proposed algorithm allows solving k-means and k-medoids problems in appropriate time. Achieved results are insignificantly less precise than the results of the evolutionary algorithms. However, new algorithm is deterministic and this fact makes its results easy for checking and interpreting by all concerned parts. The preciseness and effectiveness of the program realization of new algorithm allow solving the problem of classifying the EEE components by production batches based on the quality examination test data.

References

[1] Fedosov V V and Orlov V I 2011 Minimal necessary extent of examination of microelectronic products at inspection test stage *Izvestiya Vuzov. Priborostroenie* **54(4)** 68–62

[2] Kharchenko V S and Yurchenko Yu B 2003 Rating of fault-tolerant onboard complexes frames at usage electronic components industry *Tekhnologiya I konstruirovanie v elektronnoy apparature* **2** 3–10

[3] Kazakovtsev L A, Orlov V I, Stupina A A and Masich I S 2014 Problem of electronic components classifying *Vestnik SibGAU* **4(56)** pp 55-61

[4] Ackermann M R et al 2012 StreamKM: A Clustering Algorithm for Data Streams *J. Exp. Algorithmics* **17** 2.4:2.1-2.30

[5] Kanungo T, Mount D M, Netanyahu N S, Piatko C D, Silverman R and Wu A Y 1999 Computing nearest neighbors for moving points and applications to clustering *Proc. of the tenth annual ACM-SIAM symp. on Discrete algorithms* (Society for Industrial and Applied Mathematics) pp 931-932

[6] Kazakovtsev L A, Stupina A A and Orlov V I 2014 Modification of the genetic algorithm with greedy heuristic for continuous location and classification problems *Sistemy upravleniya i informatsionnye tekhnologii* **2(56)** 31–34

[7] Hamiter L 1991 The History of Space Quality EEE Parts in the United States *ESA Electronic Components Conf., ESTEC (Noordwijk, The Netherlands, 12–16 Nov 1990 ESA SP-313)*

[8] Kirkconnell C S et al 2014 High Efficiency Digital Cooler Electronics for Aerospace Applications *Proc. SPIE 9070, Infrared Technology and Applications XL 90702Q (June 24, 2014)*

[9] Farahani R Z and Hekmatfar M editors 2009 *Facility Location: Concepts, Models, Algorithms and Case Studies* (Berlin Heidelberg: Springer-Verlag)

[10] Kazakovtsev L A and Stupina A A 2014 Fast Genetic Algorithm with Greedy Heuristic for p-Median and k-Means *Problems IEEE 2014 6th Int. Congress on Ultra Modern Telecommunications and Control Systems and Workshops ICUMT (St.-Petersburg)* pp 702-706

[11] Kazakovtsev L A and Antamoshkin A N 2014 Genetic Algorithm wish Fast Greedy Heuristic for Clustering and Location Problems *Informatica* **38(3)** 229-240

[12] Deza M M and Deza E 2009 *Encyclopedya of Distances* (Berlin Heidelberg: Springer-Verlag)

[13] Trubin V A 1978 An efficient algorithm for the Weber problem with a rectangular metric *Kibernetika* **6** 67-70
[14] Kaufman L and Rousseeuw P J 1990 *Finding Groups in Data: an Introduction to Cluster Analysis* (New York: John Wiley & Sons)
[15] Park H S and Jun C-H 2009 A simple and fast algorithm for K-medoids clustering *Expert Systems with Applications* **36** 3336–41
[16] Kuehn A A and Hamburger M J 1963 A heuristic program for locating warehouses *Management Science* **9(4)** 643-666
[17] Lucasius C B, Dane A D and Kateman G 1993 On K-Medoid Clustering of Large Data Sets with the Aid of a Genetic Algorithm: Background, Feasibility and Comparison *Analytical Chimica Acta* **282** 647-669
[18] Zhang Q and Couloigner I 2005 A New Efficient K-Medoid Algorithm for Spatial Clustering *ICCSA 2005, LNCS 3482* pp 181–189
[19] Sheng W and Liu X 2004 A Genetic K-Medoids Clustering Algorithm *Journal of Heuristics* **12(6)** 447-466
[20] Cooper L 1963 Location-allocation problem *Oper. Res.* **11** 331–343
[21] Arthur D and Vassilvitskii S 2007 k-Means++: the Advantages of Careful Seeding *Proc. of the eighteenth annual ACM-SIAM symp. on Discrete algorithms* pp 1027–1035
[22] Mishra N, Oblinger D and Pitt L 2001 Sublinear time approximate clustering *12th SODA* pp 439–447
[23] Sun Zh, Fox G, Gu W and Li Zh 2014 A parallel clustering method combined information bottleneck theory and centroid-based clustering *The Journal of Supercomputing* **69(1)** 452–467
[24] Alp O, Erkut E and Drezner Z 2003 An Efficient Genetic Algorithm for the p-Median Problem *Annals of Operations Research* **122** 21–42
[25] Neema M N, Maniruzzaman K M and Ohgai A 2011 New Genetic Algorithms Based Approaches to Continuous p-Median Problem *Netw. Spat. Econ.* **11** 83–99
[26] Patrick M, Murphy P M and Aha D W 1994 UCI Repository of Machine Learning Databases http://www.cs.uci.edu/mlearn/mlrepository.html