Genetic Algorithm with Fast Greedy Heuristic for Clustering and Location Problems

Lev A. Kazakovtsev and Alexander N. Antamoshkin
Siberian State Aerospace University named after Academician M. F. Reshetnev
prosp. Krasnoyarskiy rabochiy, 31, Krasnoyarsk 660014, Russian Federation
E-mail: levk@bk.ru

Keywords: continuous location problem, p-median problem, genetic algorithms, discrete optimization, clustering, k-means

Received: April 4, 2014

Authors propose new genetic algorithm for solving the planar p-median location problem and k-means clustering problem. The ideas of the algorithm are based on the genetic algorithm with greedy heuristic for the p-median problem on networks and information bottleneck (IB) clustering algorithms. The proposed algorithm uses the standard k-means procedure or any other similar algorithm for local search. The efficiency of the proposed algorithm in comparison with known algorithms was proved by experiments on large-scale location and clustering problems.

Povzetek: Razvit je nov algoritem za gručenje in lokalizacijo s hitro požrešno hevristiko.

1 Introduction

The aim of a continuous location problem [18] is to determine the location of one or more new facilities in a continuum of possible locations. Main parameters of such problems are the coordinates of the facilities and distances between them [54, 19, 22]. Examples of the location problems include the location of warehouses [22], computer and communication networks, base stations of wireless networks [44, 30], statistical estimation problems [41], signal and image processing and other engineering applications. In addition, many problems of cluster analysys [21, 34, 47] can be considered as location problems [37, 32] with squared Euclidean [21, 26], Euclidean [37, 48] or other distance functions [23].

The Weber problem [52, 54] is the problem of searching for such a point that a sum of weighted Euclidean distances from this point to the given points (existing facilities which are also called "demand points" or "data vectors" in case of a clustering problem) is minimal:

$$\arg\min_{X\in\mathbb{R}^2} F(X) = \sum_{i=1}^{N} w_i L(X, A_i). \quad (1)$$

Here, $L(\cdot)$ is a distance function (norm), Euclidean in case of Weber Problem.

For solving this problem (serching for its center), we can use an iterative Weiszfeld procedure [53] or its improved modifications [51, 17]. Analogous problems with Manhattan and Chebyshev distances are well investigated [55, 50, 11]. Convergence of this algorithm is proved for various distance metrics [40].

One of possible generalizations [22, 14] of the Weber problem is the p-median problem [22] where the aim is to find optimal locations of p new points (facilities):

$$\arg\min F(X_1, ..., X_p) = \sum_{i=1}^{N} w_i \min_{j\in\{\overline{1,p}\}} L(X_j, A_i). \quad (2)$$

Here, $\{A_i | i = \overline{1,N}\}$ is a set of the demand points (data vectors), $\{X_j | j = \overline{1,p}\}$ is a set of new placed facilities, w_i is a weight coefficient of the ith demand point, $L(\cdot)$ is a distance function defined on a continuous or discrete set [24]. In this paper, we consider continuous problems in an n-dimensional space. In the simplest case, $L(\cdot)$ is Euclidean norm. In this case, the Weiszfeld procedure is implemented up to p times at aech iteration of the the iterative alternating location-allocation (ALA) method [13].

If the distance function (metric norm) is squared Euclidean (l_2^2) then the solution of the single-facility problem (1) is the mean point (centroid) [22]:

$$x_j = \sum_{i=1}^{N} w_i a_i / \sum_{i=1}^{N} w_i. \quad (3)$$

Here, we assume that $X = (x_1, ..., x_d)$, $A_i = (a_{i,1}, ..., a_{i,d})$ $\forall i = \overline{1,N}$.

The simplest and probably most popular clustering [9, 49] model is k-means [33, 34] which can be formulated as a p-median problem (2) where $w_i = 1$ $\forall i = \overline{1,N}$ and $L(\cdot)$ is squared Euclidean norm l_2:

$$L(X, Y) = \sum_{i=1}^{d} (x_1 - y_1)^2$$

where $X = (x_1, ..., x_d) \in \mathbb{R}^d$, $Y = (y_1, ..., y_d) \in \mathbb{R}^d$. Searching for the centroid takes less computational resourses than searching iteratively for the center in case of

the Weber problem and the ALA method works faster in this case.

The p-median problem with Euclidean (l_2), squared Euclidean (l_2^2) or other l_p distances [16] is a problem of global optimization: the objective function is not concave nor convex [13]. The ALA method and analogous algorithms can find one local minimum of the objective function, their result depends on the initial solution. Moreover, such global optimization problems are proved to be NP-hard [20, 2, 35] for both continuous and discrete location [45, 46, 25] which makes usage of a brute force methods impossible for large datasets. Therefore, many heuristics [39] are used to improve the obtained results. One of widely used heuristics for initial seeding is k-means++ [8]. Most popular ALA procedures for the k-means problem are based on an algorithm proposed by Lloyd [33]. The algorithm known as standard k-means procedure [34] is fast local search procedure based on Lloyd's procedure. However, many authors proposed faster methods based on this standard procedure [56, 12, 1] for datasets and continuous supplemented data streams.

Many authors propose approaches based on data sampling [38]: solving reduced problem with randomly selected part of the initial dataset and using this result as the initial solution of the ALA algorithm on the whole dataset [15, 29, 43]. Analogous approach was proposed for discrete p-median problems [6].

Many authors propose various genetic algorithms (GA) for improving the results of the local search [28, 36, 31, 42]. Most of such algorithms use evolutionary approach for recombination of the initial solution of the ALA algorithm.

Hosage and Goodchild [27] proposed the first genetic algorithm for the p-median problem. Genetic algurithms are based on the idea of recombination of elements of a set ("population") of candidate solutions called "individuals" coded by special alphabet. In [10], authors propose a genetic algorithm providing rather precise results but its convergence is slow. Alp et al. [3] proposed a quick and precise genetic algorithm with special "greedy" heuristic for solving discrete p-median problems on networks which was improved by Antamoshkin and Kazakovtsev [4]. This algorithm can be used for generating the initial solutions for the ALA algorithm [42]. The idea of the "greedy heuristic" is as follows. After selecting two "parent" solutions, new infeasible solution (a candidate solution) is composed as the union of the facility sets of the "parent" solutions. From new solution, the facilities are eliminated until the solution becomes feasible. At each step, algorithm eliminates such facility that its elimination gives minimal addition to the objective function. If this algorithm is used for the continuous p-median problem, it generates the initial solution for the ALA algorithm [42] which must be implemented at each step to estimate the result of eliminating of each facility from the candidate solution.

In this paper, we present a new genetic algorithm with floating point alphabet based on the ideas of algorithm proposed by Alp et al. [3]. Original Alp's algorithm uses integer alphabet (number of vertices of the network) in "chromosomes" (interim solutions) of the GA. Its version for planar location problems [42] uses integer alphabet for coding numbers of data vectors used as initial solutions of the ALA algorithm. In our algorithm, we use floating point alphabet. Elements of "chromosomes" of our genetic algorithm are coordinates of centers or centroids of the interim solution which are altered by steps of the ALA algorthm and eliminated until a feasible solution is obtained. Such combination of the greedy heuristic and ALA procedure allows the algorithm to get more precise results.

In case of continuous locating problems, the greedy heuristic is a computationally intensive procedure. We propose new procedure which allows eliminating sets of the centers or centroids from the candidate solution which gives multiple reduce of the running time.

2 Known algorithms

The basic idea of the alternating location-allocation ALA is recalculating the centers or centroids of the clusters and reallocating of the data vectors to the closest center or centroid:

Algorithm 1. *ALA method [28].*

Require: Set $V = (A_1, ..., A_N)$ of N data vectors in d-dimensional space, $A_1 = (a_{1,1}, ..., a_{1,d}), ..., A_N = (a_{N,1}, ..., a_{N,d})$, initial solution: a set of centers or centroid of p clusters $X_1 = (x_{1,1}, ..., x_{1,d}), ..., X_p = (x_{p,1}, ..., x_{p,d})$.

1: For each data vector, find the closest centroid:

$$C_i = \arg \min_{j=\overline{1,p}} L(A_i, X_j) \; \forall i = \overline{1, N}.$$

2: For each cluster $\mathcal{C}_j^{clust} = \{i \in \{\overline{1, N}\} | C_i = j\}$, recalculate its center or centroid X_j. In the case of Euclidean (l_2) metric, Weiszfeld procedure or its advanced modification can be used. In the case of squared Euclidean (l_2^2) metric, equation (3) is used to obtain each of d coordinates.

3: If any center or centroid was altered at Step 2 then go to Step 1.

4: Otherwise, STOP. $X_1, ..., X_p$ are local minima.

To improve the performance of Algorithm 1, recalculation of the centers or centroids are performed for the altered clusters only. In the case of Euclidean metric l_2, this allows to avoid running Weiszfeld procedure for each of the clusters at each iteration.

In case of the squared Euclidean metric l_2^2, this algorithm is called Standard k-means procedure.

The ALA methods is a local search procedure, its result depends on proper selection of the initial solution. In the simplest case, p data vectors can be randomly selected as the initial centers or centroids. A popular procedure called k-means++ for initial seeding [8] guarantees $O(\log(p))$ accuracy by proper choosing initial centers. The idea of this method is based on probability change of choosing data

vectors as the initial centers depending on distances to the closest previously chosen vectors. Analogous method for discrete location problems was proposed in [4]. The k-means++ algorithm is as follows:

Algorithm 2. *k-means++ [8].*

Require: Set $V = (A_1, ..., A_N) \in \mathbb{R}^d$, number p of clusters.

1: Initialize the probabilitiy distributions vector $P = (p_1, ..., p_N)$ with equal values (e.g. 1). Initialize the set of centroids $\chi = \emptyset$.

2: Chose one data vector X from the set of data vectors V at random using weighted probability distributions P: calculate $S = \sum_{i=1}^{N} p_i$, generate a random value $r \in [0; S)$ with the uniform distribution and use $i_{min} = \arg\min_{i \in \{\overline{1,N}\}: \sum_{j=1}^{i} p_i < r} i$. Set $\chi = \chi \cup \{A_{i_{min}}\}$.

3: For each $i \in \{\overline{1, N}\}$ set $p_i = \min_{X \in \chi} L(X, A_i)$. Here, $L(\cdot)$ is the distance metric.

4: If $|\chi| < p$ then go to Step 2.

5: Otherwise, STOP. χ is the initial set of centers or centroids.

The idea of sampling k-means [38, 15] is very simple:

Algorithm 3. *Sampling k-means.*

Require: Set $V = (A_1, ..., A_N) \in \mathbb{R}^d$, number p of clusters, parameter $s \in (0; 1)$.

1: Choose randomly s data vectors from V and form new set V_s.

2: Initialize the set of initial centers or centroids χ. To improve the results, k-means++ procedure (Algorithm algkplus) can be performed.

3: Run Algorithm 1 with the initial set χ and set of data vectors V_s. After this procedure, we have the modified set χ.

4: Run Algorithm 1 with the initial set χ for the whole set of data vectors V.

5: STOP.

In [15], authors propose a method of choosing an optimal value of the parameter s.

Sampling k-means approach, k-means++ initial seeding and other techniques improve the results of the k-means procedure, however, they do not eliminate its most important flaw: all of them perform local search. The simplest approach used for global optimization is random multistart [5]. In this case, the local search procedure runs with various randomly generated initial data. For the p-median or k-means problem, this algorithm is as follows.

Algorithm 4. *Random multistart.*

Require: Set $V = (A_1, ..., A_N) \in \mathbb{R}^d$, number p of clusters.

1: Set $F^** = +\infty$.

2: Initialize the sets of data vectors indexes $\chi : \chi \subset \{\overline{1, N}\}, |\chi_k| = p$. Uniform random generation or k-means++ procedure can be used.

3: Perform the ALA procedure with the initial solution χ and obtain a local minimum F^* of the objective function (2) and a set of corresponding centers or centroids χ^*. Instead of the "pure" ALA procedure, sampling k-means can be used in case of a large dataset.

4: If $F^{**} > F^*$ then set $F^{**} = F^*; \chi^** = \chi^*$.

5: Check the stop conditions. If they are not reached then go to Step 2.

6: Otherwise, STOP. The solution is χ^{**}.

The scheme of the genetic algorithm with greedy heuristic proposed by Alp et al. for continuous location problems is as follows [3, 42].

Algorithm 5. *GA with greedy heuristic.*

Require: Set $V = (A_1, ..., A_N) \in \mathbb{R}^d$, number p of clusters, popultion size N_p.

1: Initialize N_p sets of data vectors indexes $\chi_1, ...\chi_{N_p} : \chi_i \subset \{\overline{1, N}\}, |\chi_k| = p \; \forall k = \overline{1, N_p}$. For each $k \in \{\overline{1, N_p}\}$, calculate the fitness function. In case of the continuous p-median problem, to obtain the fitness function value for χ_k, algorithm performs the ALA procedure with initial solution χ_k and calculate

$$\mathcal{F}_k = F(\chi_k^*) = \sum_{i=1}^{N} w_i \min_{X \in \chi_k^*} L(X, A_i). \qquad (4)$$

Here, χ_k^* is the result of running the ALA procedure with the initial solution χ_k.

2: If the stop conditions are reached then go to Step 7.

3: Choose randomly two "parent" sets χ_{k_1} and χ_{k_2}, $k_1, k_2 \in \{\overline{1}, N_p\}, k_1 \neq k_2$. Running special crossover procedure with greedy heuristic, obtain "child" set of indexes χ_c. Calculate the fitness function value $\mathcal{F}_c$ in accordance with (4).

4: If $\exists k \in \{\overline{1, N_p}\} : \chi_k = \chi_c$ then go to Step 2.

5: Choose index $k_{worst} = \arg\max_{k=\overline{1,N_p}} \mathcal{F}_k$. If $\mathcal{F}_{wotst} < \mathcal{F}_c$ then go to Step 2.

6: Replace $\chi_{k_{worst}}$ with χ_c, set $\mathcal{F}_{k_{worst}} = \mathcal{F}_c$ and go to Step 2.

7: STOP. The result is a set χ_k^*, $k^* = \arg\min_{k=\overline{1,N_p}} \mathcal{F}_k$.

In the above version of this algorithm, at Steps 5 and 6, the worst solution χ_{worst} is replaced by new solution. In our experiments, we used other procedure at Step 5 (simplest tournament selection): choose randomly two indexes k_1 and k_2, $k_1 \neq k_2$; set $k_{worst} = \arg\max_{k \in \{k_1, k_2\}} \mathcal{F}_k$. This version of Step 5 gives better results.

In both random multistart and genetic algorithms, various stop conditions can be used. We used maximum running time limit.

Unlike most genetic algorithms, this method does not use any mutation procedure. However, the crossover procedure uses a special heuristic:

Algorithm 6. *Greedy crossover heuristic.*

Require: Set $V = (A_1, ..., A_N) \in \mathbb{R}^d$, number p of clusters, two "parent" sets of centers or centroids χ_{k_1} and χ_{k_2}.

1: Set $\chi_c = \chi_{k_1} \cup \chi_{k_2}$. Note that $p \le |\chi_c| \le 2p$, i.e., the candidate solution χ_c is not feasible.
2: If $|\chi_c| = p$ then STOP and return the solution χ_c.
3: Calculate $j^* = \arg\min_{j \in \chi_c} F(\chi_c \setminus \{j\})$.
4: Set $\chi_c = \chi_c \setminus \{j^*\}$.
5: Go to Step 2.

At each iteration, one index of the center or centroid is eliminated (Step 4). At Step 3, Algorithm 6 chooses the index of a center or centroid which can be eliminated with minimum change of the fitness function. To estimate the fitness function, ALA procedure must be performed. Thus, Step 3 of Algorithm 6 is computationally intensive. In case of Euclidean metric, iteratice Weiszfeld procedure must run at each iteration of the iterative ALA procedure performed $|\chi_c|$ times.

Therefore, Algorithm 6 is a computationally intensive procedure, slow for very large datasets in case of k-means problem and almost inapplicable in case of large-scale continuous p-median problems with Euclidean metric. Idea of this heuristics correlates to ideas of the information bottleneck (IB) clustering method [48]. In the IB algorithms, at the start, all data vectors form individual clusters. At each step, one cluster is eliminated, its members join other clusters. To choose such cluster, for each of them, algorithm calculates the "information loss". In the case of "geometric" clustering based on distance metrics, this loss can be estimated as the distance function increase. The computational load in case of the IB clustering allows to implement this method to small datasets only ($N < 1000$). This form of the method proposed by Alp et al. for continuous location problems is a compromise between the IB clustering simpler heuristics like traditional genetic algorithms or random multistart.

This algorithm can be used for solving a discrete p-median problem (2) with an additional condition:

$$X_j \in V \; \forall j \in \{\overline{1,p}\} \tag{5}$$

which can be used as an initial solution of the ALA method.

Algorithm 7. *Greedy crossover heuristic for initial seeding.*

Require: Set $V = (A_1, ..., A_N) \in \mathbb{R}^d$, number p of clusters, two "parent" sets of centers or centroids χ_{k_1} and χ_{k_2}.
1: Set $\chi_c = \chi_{k_1} \cup \chi_{k_2}$.
2: If $|\chi_c| = p$ then STOP and return the solution χ_c.
3: Calculate $j^* = \arg\min_{k \in \chi_c} \sum_{i=1}^{N} (\min_{j \in (\chi_c \setminus \{k\})} w_i L(A_i, A_j))$.
4: Set $\chi_c = \chi_c \setminus \{j^*\}$.
5: Go to Step 2.

In this case, the ALA method always starts from a local minimum of the discrete problem (2) with an additional constraint (5). This version of the algorithm is much faster, it gives better results than the random multistart (Algorithm 4) for most popular test datasets (see Section 4). However, such results can be improved.

We propose two modifications. One of them decreases the computational intensiveness of the algorithm, the second one improves its accuracy. Their combination makes new algorithm faster and more precise in case of large datasets.

3 Our contribution

Let us consider Steps 3 and 4 of Algorithms 6 and 7. At each iteration, Step 3 selects one index of data vectors and eliminates it from the candidate solution. Let us assume that at some kth iteration, j^*th index is eliminated and at $(k+1)$th iteration, algorithm eliminates j^{**}th index. Our first modification is based on the supposition that if A_{j^*} is distant from $A_{j^{**}}$ (i. e. $L(A_{j^*}, A_{j^{**}}) > L_{min}$, L_{min} is some constant) then the fact of eliminating or keeping j^{**}th index "almost" does not depend on the fact of elimination or keeping of j^*th index at previous iteration.

If the facts of choosing of indexes of two distant data vectors at Step 3 in two successive iterations are independent then the decisions on their eliminating (or keeping) can be made simultaneously. We propose the following modification of Steps 3 and 4.

Algorithm 8. *Fast greedy heuristic crossover: modified steps of the greedy heuristic procedure (Algorithm 6).*

3: For each $j \in \chi_c$, calculate $\delta_j = F(\chi_c \setminus \{k\})$.
4.1: Sort δ_i and select a subset $\chi_{elim} = \{e_1, ..., e_{n_\delta}\} \subset \chi_c$ of n_δ indexes with minimal values of δ_i. Value $n_\delta \in \{\overline{1, |\chi_c| - p}\}$ must be calculated in each iteration. Maximum number of the extra data elements of set χ_c must be eliminated in the first iterations and only one element in the final iterations (final iterations coincide with Algorithm 6 or 7):

$$n_\delta = \max\{[(|\chi_c| - p) * \sigma_e], 1\}. \tag{6}$$

We ran Algorithm 8 with $\sigma_e = 0.2$. Smaller values ($\sigma_e < 0.0$) convert it into Algorithm 6 and make it work slower. Big values ($\sigma_e > 0.3$) change the order of eliminating the clusters and reduce the accuracy.
4.2: From χ_{elim}, remove close data vectors. For each $j \in \{\overline{2, |\chi_{elim}|}\}$, if $\exists k \in \{\overline{1, j-1}\} : L(A_{e_j}, A_{e_k}) < L_{min}$ then remove e_j from χ_{elim}.
4.3: Set $\chi_c = \chi_c \setminus \chi_{elim}$.

Algorithm 6 performs up to p iterations. For real large datasets, computational experiments demonstrate that p or $p-1$ iterations are performed in most cases (data vectors of the "parent" solutions at Step 3 of Algorithm 5 do not coincide). In each iteration, ALA algorithm runs $|\chi_c|$ times. Thus, ALA algorithm runs up to $2p + (2p-1) + ... + 1 = 2p^2 - p + 1$ times. Popular test datasets, BIRCH 1–3 are generated for testing algorithms on problems with 100 clusters. Thus, the ALA algorithm must run up to 19901 times.

Depending on parameter L_{min}, in each iteration of Algorithm 8 eliminates up to $\sigma_e \cdot p$ members from χ_c. If L_{min} is big and $\sigma_e = 0.2$, in the first iteration, ALA runs $2p$ times, in the second iteration $[1.8p]$ times, then $[1.64p]$, $[1.512p]$ times etc. In case of 100 clusters and big L_{min}, the ALA procedure runs $200 + 180 + 164 + 152 + 142 + 134 + 128 + 123 + 118 + 116 + 113 + 111 + 109 + 108 + 107 + 106 + 105 + 104 + 103 + 102 + 101 = 2626$ times only. Taking into account computational intenseness or the ALA procedure such as standard k-means algorithm which is estimated $O(N^{34}p^3 4\log^4(N)/\sigma^6)$ in case of independently perturbed data vectors by a normal distribution with variance σ^2 [7], reducing number of runs of the local search procedure is crucial in case of large-scale problems.

Step 3 of Algorithm 7 can be modified as follows.

Algorithm 9. *Fast greedy heuristic crossover for initial seeding: modified steps of the greedy heuristic procedure (Algorithm 7).*

3: For each $j \in \chi_c$, calculate $\delta_j = \sum_{i=1}^{N} (\min_{j \in (\chi_c \setminus \{k\})} w_i L(A_i, A_j))$.

4.1: Sort δ_i and select a subset $\chi_{elim} = \{e_1, ..., e_{n_\delta}\} \subset \chi_c$ of n_δ indexes with minimal values of δ_i.

4.2: For each $j \in \{\overline{2, |\chi_{elim}|}\}$, if $\exists k \in \{\overline{1, j-1}\} : L(A_{e_j}, A_{e_k}) < L_{min}$ then remove e_j from χ_{elim}.

4.3: Set $\chi_c = \chi_c \setminus \chi_{elim}$.

The aim of Step 4.2 of Algorithm 8 is to hold the order of elimination of the clusters provided by Algorithms 6 or 7. In Fig. 1, two cases of running Algorithm 8 are shown. Let us assume that $p = 4$ and distances between the centers of clusters 1 and 3, 3 and 4, 1 and 4, 6 and 7 are less than L_{min}. Let us assume that parameter σ_e allows eliminating up to 3 clusters simultaneously in the 1st iteration. After Step 3 of Algorithm 8 and sorting δ_i, we have a sequence of clusters $4,\ 3,\ 6, \ldots$. If Step 4.2 is included in Algorithm 8 then only one of clusters 1, 3 and 4 can be removed in the 1st iteration (Case A). Thus, only two clusters (4 and 7) are eliminated in the 1st iteration. If we remove Step 4.2 from our algorithm or assign big value to L_{min} then the simultaneous elimination of clusters 3 and 4 is allowed (Case B) which gives worse value of the squared distances sum. If the original Algorithm 7 runs, it eliminates cluster 4 first, then cluster 6. In its 3rd iteration, Algorithm 7 eliminates cluster 1 and we have the set of clusters shown in Fig. 1, Case A after two iterations which coincides with the result of Algorithm 8.

Algorithm 6 starts the ALA procedure many times, it is a precise but slow method. Having included Algorithm 8 into Algorithm 6, we reduce the number of starts of the ALA procedure, however, as explained above, at least 2626 starts of the local search algorithm in each iteration of the genetic algorithm in case of 100 clusters make using this method impossible for very a large dataset, especially for the Euclidean metric. Algorithm 7 optimizes the fitness function calculated for the initial seeding of the ALA procedure. This approach is fast, however, an optimal value of the fitness function for the initial seeding does not guarantee its optimal value for the final result of the ALA procedure.

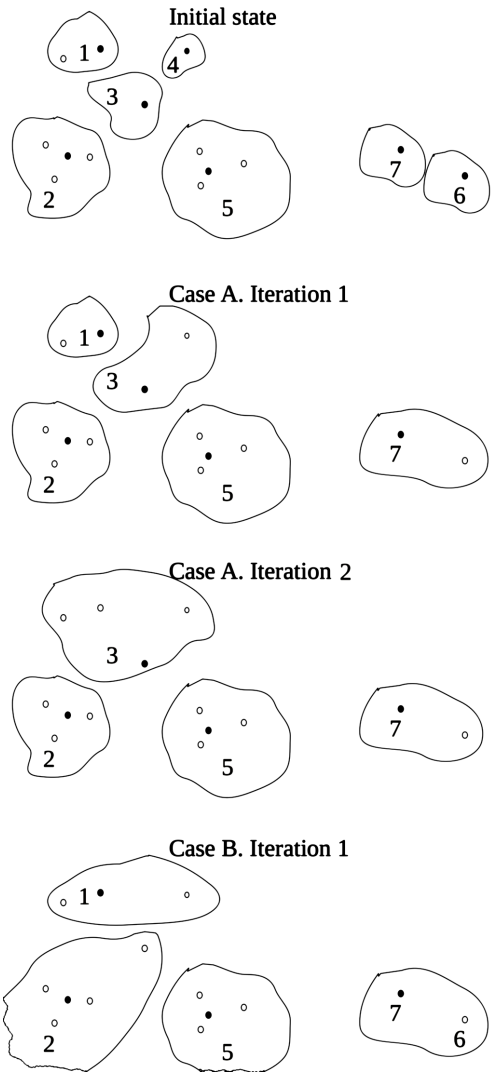

Figure 1: Succeeding and simultaneous elimination of clusters.

We propose a compromise version of two algorithms which implements one step of the ALA procedure after each elimination of the clusters. Since the result of the ALA procedure does not coincide with the data vectors A_i (in general), using integer numbers as the alphabet of the GA (i.e. for coding the solutions forming the population of the GA) is impossible and we use real vectors (coordinates of the interim solutions of the ALA procedure) for coding the solutions in the population of the GA. The whole algorithm is as follows.

Algorithm 10. *GA with greedy heuristic and floating point alphabet.*

Require: Set $V = (A_1, ..., A_N) \in \mathbb{R}^d$, number p of clusters, population size N_p.

1: Initialize N_p sets of coordinates $\chi_1, ..., \chi_{N_p}$: $chi_i \subset \mathbb{R}^d$, $|\chi_k| = p \ \forall k = \overline{1, N_p}$ with solutions of the ALA algorithm initialized by the k-means++ procedure (Algorithm 2). Thus, each χ_i is a local minimum of (2). Store corresponding values of the function 2 to array $\mathcal{F}_1, ..., \mathcal{F}_{N_p}$.

2: If the stop conditions are reached then go to Step 7.

3: Choose randomly two "parent" sets χ_{k_1} and χ_{k_2}, $k_1, k_2 \in \{\overline{1}, N_p\}$, $k_1 \neq k_2$. Running Algorithm 11, obtain "child" set of coordinates χ_c which is a local minimum of (2). Store the value of (2) to $\mathcal{F}_c$.

4: If $\exists k \in \{\overline{1, N_p}\} : \chi_k = \chi_c$ then go to Step 2.

5: Choose index $k_{worst} = \arg\max_{k=\overline{1,N_p}} \mathcal{F}_k$. If $\mathcal{F}_{wotst} < \mathcal{F}_c$ then go to Step 2.

6: Choose randomly two indexes k_1 and k_2, $k_1 \neq k_2$; set $k_{worst} = \arg\max_{k \in \{k_1, k_2\}} \mathcal{F}_k$.

6: Replace $\chi_{k_{worst}}$ with χ_c, set $\mathcal{F}_{k_{worst}} = \mathcal{F}_c$ and go to Step 2.

7: STOP. The result is a set χ^*_k, $k^* = \arg\min_{k=\overline{1,N_p}} \mathcal{F}_k$.

The greedy heuristic procedure is modified as follows.

Algorithm 11. *Greedy crossover heuristic with floating point alphabet.*

Require: Set $V = (A_1, ..., A_N) \in \mathbb{R}^d$, number p of clusters, two "parent" sets of centers or centroids χ_{k_1} and χ_{k_2}, parameters σ_e and L_{min}.

1: Set $\chi_c = \chi_{k_1} \cup \chi_{k_2}$. Run the ALA procedure for $|\chi_c|$ clusters starting from χ_c. Store its result to χ_c.

2: If $|\chi_c| = p$ then run the ALA procedure with the initial solution χ_c, then STOP and return its result.

2.1: Calculate the distances from each data vector to the closest element of χ_c.

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

Assign each data vector to the corresponding cluster with its center in an element of χ_c.

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

Calculate the distances from each data vector to the second closest element of χ_c.

$$D_i = \min_{Y \in (\chi_c \setminus \{C_i\})} L(Y, A_i).$$

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

4.1: Calculate n_δ in accordance with (6). Sort δ_X and select a subset $\chi_{elim} = \{X_1, ..., X_{n_\delta}\} \subset \chi_c$ of n_δ coordinates with minimal values of δ_X.

4.2: For each $j \in \{\overline{2, |\chi_{elim}|}\}$, if $\exists k \in \{\overline{1, j-1}\} : L(X_j, X_k) < L_{min}$ then remove X_j from χ_{elim}.

4.3: Set $\chi_c = \chi_c \setminus \chi_{elim}$.

4.4: Reassign data vectors to the closest centers or centroids.

$$C^*_i = \arg\min_{X \in \chi_c} L(X, A_i) \ \forall i = \overline{1, N}.$$

For each $X \in \chi_c$, if $\exists i \in \{\overline{1, N}\} : C_i = X$ and $C^* i \neq X$ then recalculate center or centroid X^* of the cluster $\mathcal{C}^{clust}_X = \{A_i | C^*_i = X, i = \overline{1, N}\}$. Set $\chi_c = (\chi_c \setminus \{X^*\}) \cup \{X\}$.

5: Go to Step 2.

An important parameter of Algorithms 8 and 11 is L_{min}. Performed series of experiments on various data, we propose the following method of its determining for each pair of centers or centroids X_j and X_k (see Step 4.2 of Algorithm 11):

$$L_{min} = \min_{X \in \chi_c} \{\max\{L(X, X_j), L(X, X_k)\}\}.$$

We ran this algorithm with large datasets and proved its efficiency experimentally.

4 Computational experiments

4.1 Datasets and computing facilities

For testing purposes, we used real data and generated datasets collected by Speech and Image Processing Unit of School of Computing of University of Eastern Finland[1] and UCI Machine Learning Repository[2]. Other authors use such problems for their experiments [56, 1, 43]. Number of data vectors N varies from 150 (classical Iris plant problem) to 581013 (Cover type dataset), number of dimensions d varies from 2 to 54, number of clusters from 3 to 1000. In addition, we used specially generated datasets for p-median problems (uniformly distributed data vecrots in $\mathbb{R}^2$, each coordinate in interval $[0; 10)$ with uniformly distributed weights in range $[0; 10)$).

Computational experiments were performed for problems with Euclidean (l_2) and squared Euclidean (l_2^2) distances (p-median and k-means problems, correspondingly).

For our experiments, we used a computer Depo X8Sti (6-core CPU Xeon X5650 2.67 GHz, 12Gb RAM), hyperthreading disabled and ifort compiler with full optimization and implicit parallelism (option -O3).

For algorithms comparison purposes, we ran each algorithm with each of datasets 30 times.

4.2 Algorithm parameters tuning

An important parameter of the genetic algorithm is number of individuals (candidate solutions) N_p in its population (population size). Running Algorithm 10 for the generated datasets ($d = 2$, $N = 1000$ and $N = 10000$, $p = 10$ and

[1] http://cs.joensuu.fi/sipu/datasets/

[2] https://archive.ics.uci.edu/ml/datasets.html

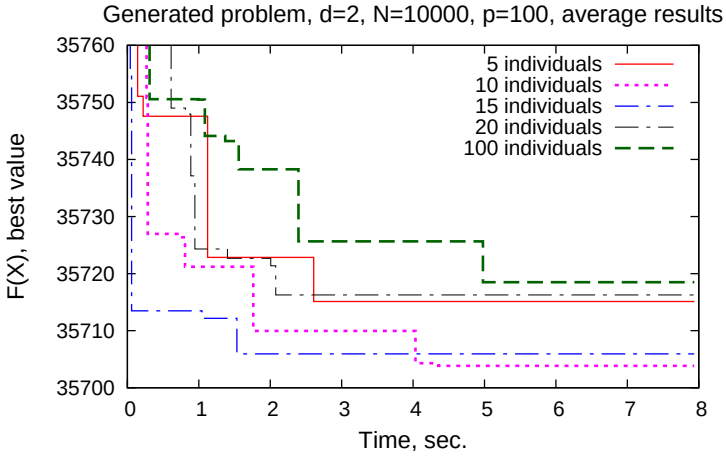

Figure 2: Results of Algorithm 10 with various population sizes.

$p = 100$) and real datasets (MissAmerica1 with $p = 100$) show that large populations ($N_p > 50$) slow down the convergence. Analogously, running with very small populations ($N_p < 10$) decrease the accuracy.

Results of running our algorithm for a generated problem with squared Euclidean metric are shown in Fig. 2. In this diagram, we fixed the best results achieved by the algorithms and the spent time after each improvement of the results in a special array. This diagram shows the average or worst results of 30 starts of the algorithms.

Experiments show that a population of 10–25 individuals is optimal for all tested problems for all greedy crossover heuristics considered in this paper (Algorithms 6, 7, 8 and 11). There is no obviois relation between d and N_p, p and N_p nor N and N_p. In all experiments below, we used $N_p = 15$.

4.3 Numerical results

For all datasets, we ran the genetic algorithm with greedy heuristic (Algorithm 5) with various crossover heuristics (Step 3 of Algorithm 5): its original version proposed by Alp et al. [3, 42] (Algorithm 6), its version for initial cluster centers seeding only (Algorithm 7), our modifications allowing elimination of many cluster centers in one step (Algorithm 8) and our new genetic algorithm with floating point alphabet (Algorithm 11).

Results for each of datasets are shown in Tables 1 and 2.

We used the sampling k-means procedure (Algorithm 3) for datasets with $N \geq 10000$ as the ALA procedure at Step 1 of Algorithms 5, 10 and 11. For smaller datasets, we ran all algorithms without sampling. However, running algorithms without sampling k-means procedure for large datasets equally delays the genetic algorithm with all considered greedy crossover heuristics.

Computation process with each of the algorithms was performed 30 times. Time limit shown in the first column was used as the stop condition. Value of this maximum running time was chosen so that adding equal additional time does not allow to obtain better results in case of using the original greedy crossover heuristic for initial seeding (Algorithm 7) in at least 27 attempts of 30. In addition, for the problems listed in Table 1, we fixed the average time needed for achieving the average result of the original genetic algorithm with greedy crossover heuristic (Algorithm 5 + Algorithm 6, see [3, 42]). For more complex problems listed in Table 2 where the original greedy crossover procedure is inapplicable due to huge computational complexity, we fixed the average time needed for achieving the average result of the original genetic algorithm with greedy crossover heuristic applied for optimizing the fitness function value of the initial dataset of the ALA procedure (Algorithm 5 + Algorithm 7).

Algorithm 5 with the original greedy crossover heuristic (Algorithm 6) proposed by Alp et al. [3, 42] shows excellent results for comparatively small datasets (see Table 1). For the least complex problems ("Iris" dataset), using the algorithm proposed in this article (Algorithm 10, Problems 1 and 3) reduces the accuracy of the solution in comparison with the original algorithm of Alp et al. [3, 42] (Algorithms 5 and 6). For larger datasets, new algorithm (Algorithm 10+Algorithm 11) is faster and more precise.

Moreover, using the original greedy crossover heuristic is impossible for large datasets (for all larger gatasets with $p > 30$, $N \geq 10000$) due to very intensive computation at each iteration. For such datasets, we used the algorithm of Alp et al. applied for optimizing the fitness function value of the initial dataset of the ALA procedure (Algorithms 5 and 7) for comparison purposes. In this case, for all solved large-scale test problems with both Euclidean (l_2, planar p-median problem) and squared Euclidean (l_2^2, k-means problem) metrics, our Algorithm 10 with floating point alphabet and modified greedy crossover heuristic (Algorithm 11) works faster and gives more precise results than Algorithm 5 with greedy heuristic implemented for the initial seeding only (Algorithm 7, [3, 42]).

Table 1: Results of running new Algorithm 11 and original genetic algorithm with greedy crossover heuristic.

Dataset, and its parameters, time limit	Distance	Method	Avgerage result	Average time needed for reaching result of the original method, sec.	Worst result	Avg. speedup (new vs. original method)
Iris ($n = 150, d = 4$, $p = 3$), 100 sec.	l_2^2	Original	$1.40026039044 \cdot 10^{14}$	0.0096	$1.40026039044 \cdot 10^{14}$	
		ALA mult.	$1.40026039044 \cdot 10^{14}$	0.0103	$1.40026039044 \cdot 10^{14}$	
		New	$1.400262 \cdot 10^{14}$	-	$1.4002858 \cdot 10^{14}$	-
Iris ($n = 150, d = 4$, $p = 10$), 100 sec.	l_2^2	Original	46916083985700	2.4	46916083985700	
		ALA mult.	46917584582154	-	46935815209300	
		New	46916083985700	2.5	46916083985700	-
				-		-
MissAmerica1 ($n = 6480, d = 16$, $p = 30$), 1500 sec.	l_2^2	Original	105571815.061	603	105663081.95	
		ALA mult.	105714622.427	-	106178506.965	
		New	105440299.602	13.8	105440299.601	43.69
				-		-
Europe ($n = 169309$, $d = 2, p = 10$), 1500 sec.	l_2^2	Original	1099348562.46	1050.8	1099355026.03	
		ALA mult.	1099345009.09	15.6	1099345033.08	
		New	1099345067.99	123.8	1099345210.55	8.48
				-		-

Note:"Original" algorithm is Algorithm 5 with original greedy crossover heuristic (Algorithm 6),
"ALA mult." algorithm is multiple start of the ALA procedure (Algorithm 4),
"New" algorithm is the genetic algorithm with floating point alphabet (Algorithm 10 with Algorithm 11 as the greedy crossover procedure).

To illustrate the dynamics of the solving process, we present the timing diagrams which show the average results of 30 runs of each algorithm for various datasets in Fig. 3 and 4. Diagrams show that new algorithm with floating point alphabet allows to increase the accuracy at early stages of the computation process in comparison with known methods which allows to use it for obtaining quick approximate solutions. In addition, results of the fast greedy heuristic (Algorithm 8) are shown in the diagrams. Using this heuristic without other modifications to the genetic algorithm can reduce the accuracy of the results.

5 Conclusion

New genetic algorithm based on ideas of the p-median genetic algorithm with greedy heuristic and two original procedures can be used for fast and precise solving the large-scale p-median and k-means problems. For the least complex problems, the results of our method are less precise than the original GA with greedy heuristic proposed by Alp et al. However, new algorithm provides more precise results in appropriate time for the large-scale problems.

References

[1] M. R. Ackermann et al. (2012) StreamKM: A Clustering Algorithm for Data Streams, *J. Exp. Algorithmics*, Vol 17, Article 2.4 (May 2012), DOI: 10.1145/2133803.2184450

[2] D. Aloise, A. Deshpande, P. Hansen, P. Popat (2009) NP-Hardness of Euclidean Sum-of-Squares Clustering, *Machine Learning*, Vol. 75, pp. 245–249, DOI: 10.1007/s10994-009-5103-0

[3] O. Alp, E. Erkut and Z. Drezner (2003) An Efficient Genetic Algorithm for the p-Median Problem, *Annals of Operations Research*, Vol.122, Issue 1–4, pp. 21–42, doi 10.1023/A:1026130003508

[4] A. Antamoshkin, L. Kazakovtsev (2013) Random Search Algorithm for the p-Median Problem, *Informatica (Ljubljana)*, Vol. 37, No. 3, pp. 267–278

[5] A. Antamoshkin, H. P. Schwefel, A. Toern, G. Yin, A. Zhilinskas (1993) *Systems Analysis, Design and Optimization. An Introduction*, Krasnoyarsk, Ofset

[6] P. Avella, M. Boccia, S. Salerno and I. Vasilyev (2012) An Aggregation Heuristic for Large Scale p-median Problem, *Computers & Operations Research*, 39 (7), pp. 1625–1632, doi 10.1016/j.cor.2011.09.016

Table 2: Results of running new Algorithm 11 and original genetic algorithm with greedy crossover heuristic used for initial seeding.

Dataset its parameters, time limit	Distance	Method	Avgerage result	Average time needed for reaching result of the original method, sec.	Worst result	Avg. speedup (new vs. original method for init. seeding)
Europe ($n = 169309$, $d = 2, p = 100$), 1200 sec.	l_2	Orig. init. ALA mult. New	400370576 400755480 400345100	397.6 - 193.6	400904292 401007437 400595350	- 2.05
Generic ($n = 10000$, $d = 2, p = 100$), 300 sec.	l_2	Orig. init. ALA mult. New	85318.44 85374.62 85167.01	47.65 - 0.895	85482.67 86114.83 85179.72	- 53.24
Europe ($n = 169309$, $d = 2, p = 100$), 1200 sec.	l_2^2	Orig. init. ALA mult. New	$2.2767 \cdot 10^{12}$ $2.3039 \cdot 10^{12}$ $2.2752 \cdot 10^{12}$	557.6 - 143.1	$2.2933 \cdot 10^{12}$ $2.3111 \cdot 10^{12}$ $2.2862 \cdot 10^{12}$	- 3.89
BIRCH1 ($n = 100000$, $d = 2, n = 100$), 120 sec.	l_2^2	Orig. init. ALA mult. New	$9.277283 \cdot 10^{13}$ $9.746921 \cdot 10^{13}$ $9.277282 \cdot 10^{13}$	46.08 - 31.56	$9.277287 \cdot 10^{13}$ $9.276386 \cdot 10^{13}$ $9.274231 \cdot 10^{13}$	- 1.46
BIRCH3 ($n = 100000$, $d = 2, p = 1000$), 2400 sec.	l_2^2	Orig. init. ALA Mult. New	$4.08652 \cdot 10^{12}$ $4.16053 \cdot 10^{12}$ $4.02040 \cdot 10^{12}$	802.8 - 8.81	$4.105231 \cdot 10^{12}$ $4.162683 \cdot 10^{12}$ $4.022190 \cdot 10^{12}$	- 91.12
MissAmerica2 ($n = 6480$, $d = 16, p = 100$), 300 sec.	l_2^2	Orig.init. ALA Mult. New	80264286 81869326 79837119	85.8 - 0.752	81039148 82316364 80147971	- 114.1
CoverType ($n = 581013$, $d = 54, p = 100$), 2400 sec.	l_2^2	Orig.init. ALA Mult. New	3122934.7 3163271.9 3069213.6	905.4 - 53.1	3146107.4 3182076.8 3072299.5	- 17.05

Note: "Orig. init." algorithm is Algorithm 5 with original greedy crossover heuristic (Algorithm 7),
"ALA mult." algorithm is multiple start of the ALA procedure (Algorithm 4),
"New" algorithm is the genetic algorithm with floating point alphabet (Algorithm 10 with Algorithm 11 as the greedy crossover procedure).

[7] D. Arthur, B. Manthey and H. Roglin (2009) k-Means Has Polynomial Smoothed Complexity, *in Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '09)*, IEEE Computer Society, Washington, DC, USA, pp. 405–414 DOI: 10.1109/FOCS.2009.14

[8] D. Arthur and S. Vassilvitskii (2007) k-Means++: The Advantages of Careful Seeding, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete algorithms*, ser. SODA '07, pp. 1027–1035

[9] K. Bailey (1994) Numerical Taxonomy and Cluster Analysis, in: Typologies and Taxonomies, Sage Pubns, DOI: 10.4135/9781412986397

[10] B. Bozkaya, J. Zhang and E. Erkut (2002) A Genetic Algorithm for the p-Median Problem, *in Z. Drezner and H. Hamacher (eds.), Facility Location: Applications and Theory*, Springer

[11] A. V. Cabot, R. L. Francis and M. A. Stary (1970) A Network Flow Solution to a Rectilinear Distance Facility Location roblem, *American Institute of Industrial Engineers Transactions*, 2, pp. 132–141

[12] L. O'Callaghan, A. Meyerson, R. Motwani, N. Mishra and S. Guha (2002) Streaming-Data Algorithms for High-Quality Clustering, *Data Engineering, 2002. Proceedings. 18th In-*

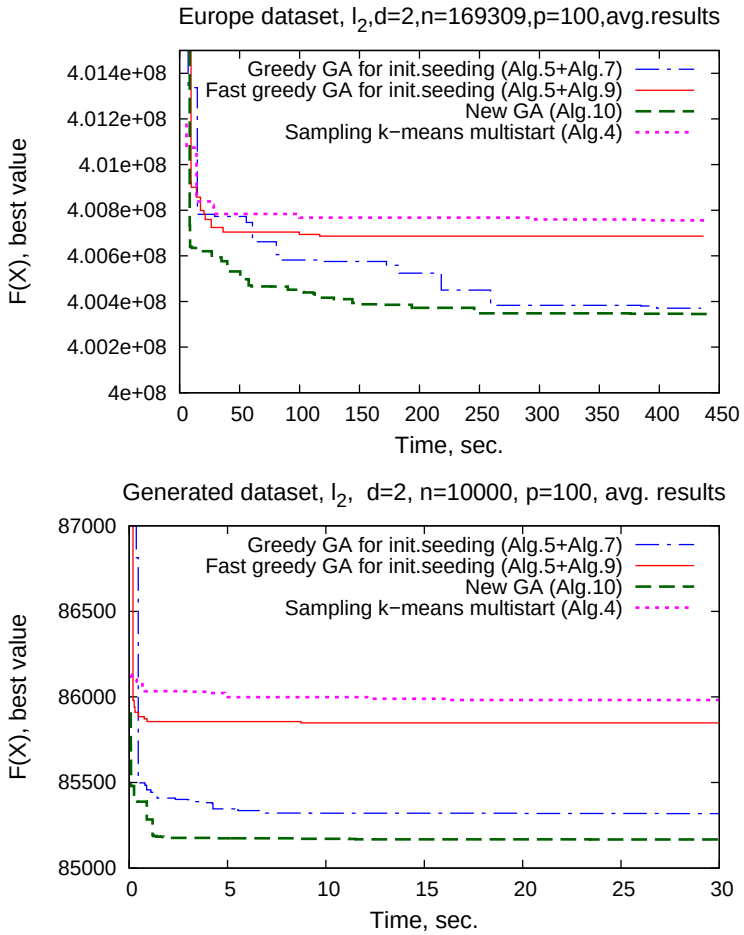

Figure 3: Average results of running new GA in comparison with other algorithms for $p-$median problems.

ternational Conference on, pp. 685–694 DOI: 10.1109/ICDE.2002.994785

[13] L. Cooper (1963) Location-allocation problem, *Oper Res*, vol. 11, pp. 331–343

[14] L. Cooper (1968) An Extension of the Generalized Weber Problem, *Journal of Regional Science*, Vol. 8, Issue 2, pp.181–197

[15] A. Czumaj and C. Sohler (2004) Sublinear-Time Approximation for Clustering Via Random Sampling, *in Automata, Languages and Programming, Lecture Notes in Computer Science*, Vol. 3142, Springer Berlin Heidelberg, pp. 396–407, DOI: 10.1007/978-3-540-27836-8_35

[16] M. M. Deza, E. Deza (2013) Metrics on Normed Structures, *Encyclopedia of Distances,* Springer Berlin Heidelberg, pp.89–99, DOI: 10.1007/978-3-642-30958-8_5

[17] Z. Drezner (2013) The Fortified Weiszfeld Algorithm for Solving the Weber Problem, *IMA Journal of Management Mathematics*, published online. DOI: 10.1093/imaman/dpt019

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

[19] Z. Drezner and G. O. Wesolowsky (1978) A Trajectory Method for the Optimization of the Multifacility Location Problem with lp Distances, *Management Science*, 24, pp.1507-1514

[20] P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay (1999) Clustering Large Graphs via the Singular Value Decomposition, *Machine learning*, vol. 56(1-3), pp. 9–33

[21] B. S. Duran, P. L. Odell (1977) *Cluster Analysys: a Survey*, Springer-Verlag Berlin–Heidelberg–New York

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

[23] V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell and J. French (1999) Clustering large datasets in arbitrary metric spaces, *Proc. 15th Int. Conf. Data Engineering*, pp. 502–511

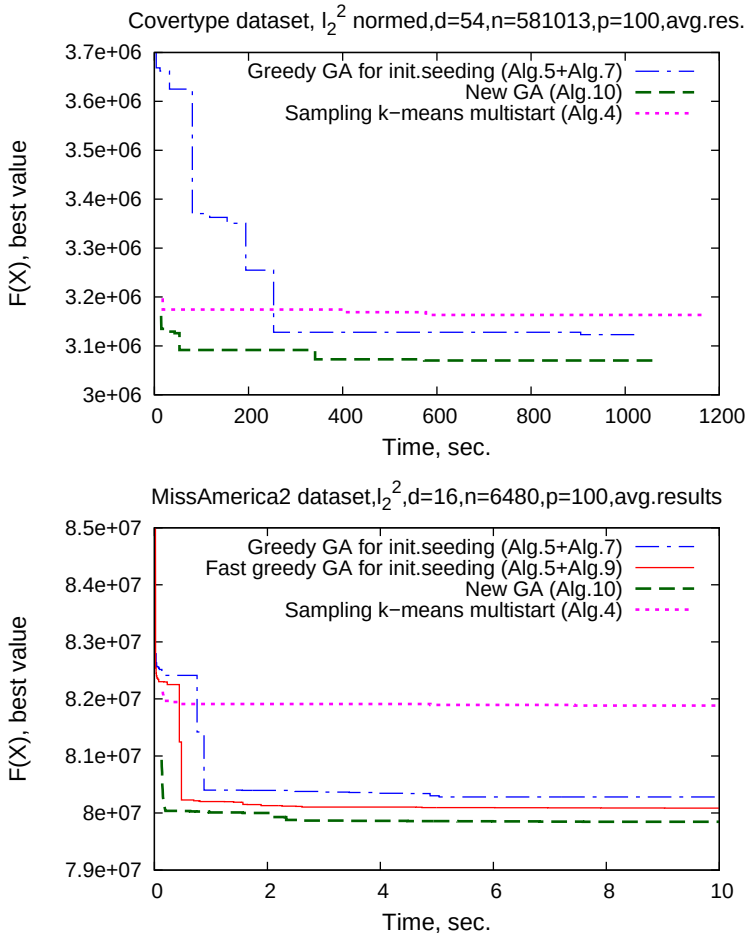

Figure 4: Average results of running new GA in comparison with other algorithms for k-means problems.

[24] S. L. Hakimi (1964) Optimum Locations Of Switching Centers and the Absolute Centers and Medians of a Graph, *Operations Research*, 12(3), pp. 450-459

[25] P. Hansen, J. Brimberg, D. Urošević, N. Mladenović (2009) Solving large p-median clustering problems by primal-dual variable neighborhood search, *Data Mining and Knowledge Discovery*, vol. 19, issue 3, pp 351–375

[26] P. Hansen, E. Ngai, B. Cheung, N. Mladenović (2005) Analysis of Global k-Means, an Incremental Heuristic for Minimum Sum of Squares, *Journal of Classification*, vol. 22, issue 3, pp 287–310

[27] C. M. Hosage and M. F. Goodchild (1986) Discrete Space Location–Allocation Solutions from Genetic Algorithms, *Annals of Operations Research* 6, 35–46.

[28] C. R. Houck, J. A. Joines and M. G. Kay (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

[29] R. Jaiswal, A. Kumar and S. Sen (2013) A Simple D2-Sampling Based PTAS for k-Means and Other Clustering Problems, *Algorithmica*, DOI: 10.1007/s00453-013-9833-9

[30] L. Kazakovtsev (2013) Wireless Coverage Optimization Based on Data Provided by Built-In Measurement Tools, *WASJ*, vol.22, spl. issue Tech. and Technol., pp. 8–15

[31] K. Krishna, M. Narasimha Murty (1999) Genetic K-Means Algorithm, *IEEE Transactions on Systems, Man, and Cybernetics – part B: Cybernetics*, Vol. 29, No. 3, pp. 433–439

[32] K. Liao and D. Guo (2008) A Clustering-Based Approach to the Capacitated Facility Location Problem, *Transactions in GIS*, vol. 12 (3), pp.323–339

[33] S. P. Lloyd (1982) Least Squares Quantization in PCM, *IEEE Transactions on Information Theory*, Vol. 28, pp. 129–137

[34] J. B. MacQueen (1967) Some Methods of Classification and Analysis of Multivariate Observations, *Proceedings of the 5th Berkley Symposiom on Mathematical Statistics and Probability*, vol. 1, pp. 281–297

[35] S. Masuyama, T. Ibaraki and T. Hasegawa (1981) The Computational Complexity of the m-Center Problems on the Plane, *The Transactions of the Institute of Electronics and Comunication Engineers of Japan*, 64E, pp. 57–64

[36] U. Maulik, S. Bandyopadhyay (2000) Genetic Algorithm-Based Clustering Technique, *Pattern Recognition,* Vol. 33, pp. 1455–1465

[37] L. A. A. Meira and F K. Miyazawa (2008) A Continuous Facility Location Problem and Its Application to a Clustering Problem, *Proceedings of the 2008 ACM symposium on Applied computing (SAC '08)*. ACM, New York, USA, pp.1826–1831. doi: 10.1145/1363686.1364126

[38] N. Mishra, D. Oblinger and L. Pitt (2001) Sublinear time approximate clustering, *12th SODA*, pp. 439–447

[39] N. Mladenović, J. Brimberg, P. Hansen, J. A. Moreno-Perez (2007) The p-median problem: A survey of metaheuristic approaches, *European Journal of Operational Research*, Vol. 179, issue 3, pp.927–939

[40] J. G. Morris (1981) Convergence of the Weiszfeld Algorithm for Weber Problem Using a Generalized "Distance" Function, *Operations Research*, vol. 29 no. 1, pp. 37–48

[41] I. A. Osinuga and O. N. Bamigbola (2007) On the Minimum Norm Solution to the Weber Problem, *SAMSA Conference proceedings*, pp. 27–30

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

[43] P. Phoungphol and Y. Zhang (2011) Sample Size Estimation with High Confidence for Large Scale Clustering, it Proceedings of the 3rd International Conference on Intelligent Computing and Intelligent Systems, http://www.cs.gsu.edu/ pphoungphol1/paper/icis2011.pdf

[44] A. W. Reza, K. Dimyati, K. A. Noordin, A. S. M. Z. Kausar, Md. S. Sarker (2012) A Comprehensive Study of Optimization Algorithm for Wireless Coverage in Indoor Area, *Optimization Letters*, September 2012, published online, doi 10.1007/s11590-012-0543-z, http://link.springer.com/article/10.1007 %2Fs11590-012-0543-z?LI=true

[45] M. G. C. Resende (2008) Metaheuristic hybridization with Greedy Randomized Adaptive Search Procedures, *in TutORials in Operations Research, Zhi-Long Chen and S. Raghavan (Eds.)*, INFORMS, pp. 295–319

[46] M. G. C. Resende, C. C. Ribeiro, F. Glover and R. Marti (2010) Scatter search and path-relinking: Fundamentals, advances, and applications, *Handbook of Metaheuristics, 2nd Edition, M. Gendreau and J.-Y. Potvin, Eds.*, Springer pp. 87–107

[47] X. Rui and D. Wunsch (2005) Survey of Clustering Algorithms, *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp.645–678, doi: 10.1109/TNN.2005.845141

[48] S. Still, W. Bialek and L. Bottou (2004) Geometric Clustering using the Information Bottleneck method, in: *Advances In Neural Information Processing Systems 16* Eds.: S. Thrun,L. Saul, and B. Scholkopf, MIT Press, Cambridge, MA.

[49] P.-N. Tan, M. Steinbach, V. Kumar (2006) Cluster Analysis: Basic Concepts and Algorithms, Chapter 8 in: *Introduction to Data Mining*, Addison-Wesley, pp. 487–567

[50] V. A. Trubin (1978) Effective algorithm for the Weber problem with a rectangular metric, *Cybernetics and Systems Analysis*, **14(6)**, DOI:10.1007/BF01070282, Translated from Kibernetika, **6** (November-December 1978), pp. 67–70.

[51] Y. Vardi and C.-H. Zhang (2001) A Modified Weiszfeld Algorithm for the Fermat-Weber Location Problem, *Math. Program., Ser. A*, vol. 90: pp. 559–566, DOI: 10.1007/s101070100222

[52] A. Weber (1922) *Über den Standort der Industrien, Erster Teil: Reine Theorie des Standortes*, Tübingen, Mohr

[53] E. Weiszfeld (1937) Sur le point sur lequel la somme des distances de n points donnes est minimum, *Tohoku Mathematical Journal*, **43** no.1, pp. 335–386.

[54] G. Wesolowsky (1992) The Weber problem: History and perspectives, *Location Science*, **1**, pp. 5–23.

[55] G. O. Wesolowsky and R. F. Love (1972) A nonlinear Approximation Method for Solving a Generalized Rectangular Distance Weber Problem, *Management Science*, vol. 18 no. 11, pp. 656–663

[56] T. Zhang, R. Ramakrishnan and M. Livny (1996) BIRCH: An Efficient Data Clustering Method for Very Large Databases, *Proceedings of the 1996 ACM SIGMOD international conference on Management of data (SIGMOD '96)*, ACM, New York, NY, USA, pp. 103–114, DOI: 10.1145/233269.233324