A Genetic Algorithm Implementation of the Fuzzy Least Trimmed Squares Clustering

Amit Banerjee and Sushil J. Louis, *Member, IEEE*

Abstract— **This paper describes a new approach to finding a global solution for the fuzzy least trimmed squares clustering. The least trimmed squares (LTS) estimator is known to be a high breakdown estimator, in both regression and clustering. From the point of view of implementation, the feasible solution algorithm is one of the few known techniques that guarantees a global solution for the LTS estimator. The feasible solution algorithm divides a noisy data set into two parts – the non-noisy retained set and the noisy trimmed set, by implementing a pairwise swap of datum between the two sets until a least squares estimator provides the best fit on the retained set. We present a novel genetic algorithm-based implementation of the feasible solution algorithm for fuzzy least trimmed squares clustering, and also substantiate the efficacy of our method by three examples.**

I. Introduction

Clustering procedures produce a partition of data into clusters, with similar entities grouped in a cluster and dissimilar entities in different clusters. The K-Means class of clustering algorithms generate non-overlapping *k*-partitions of the data by minimizing the least squares (LS) residual within every cluster. Fuzzy *c*-Means (FCM) is the fuzzy extension of the ever popular K-Means and generates overlapping clusters where every entity is shared (with varying degrees of membership) across all the clusters. However, classical non-robust estimators like the LS residual minimization techniques fail miserably when the data set is contaminated with noise and/or outliers. The breakdown point for such estimators – the minimum fraction of outliers present in the data set to cause the estimator to be arbitrarily wrong – approaches zero when the sample size gets larger. To be useful in practice, when most of the data that analysts deal with are inherently noisy, clustering techniques should be robust with an ability to generate natural clusters even in the presence of outliers.

In the field of regression statistics, several robust estimators, such as the L_1 estimator, M-estimator, generalized R-estimators and L-estimators, have been proposed and have been shown to be resistant to noise. However, they are all low breakdown regression estimators which severely restricts their usability. In fact, these estimators can withstand a large number of outliers in the dependent variable but break down easily in the presence of leverage points. The first high breakdown (~50%) regression estimator proposed was the Repeated-Median estimator [1], following which Rousseeuw [2] introduced his least median of squares (LMS) estimator. More recently, the least trimmed squares (LTS) and the re-weighted least trimmed squares (RLS) estimator [3] have been proposed. These estimators belong to a family of so-called S-estimators. Rousseeuw was also the first to extend the idea of robust estimators to clustering with his K-Mediod algorithm [3], which used the absolute residual (L_1) instead of the squared residual minimization in LS-based techniques. The Robust *c*-Prototypes (RCP) [4] and the robust Fuzzy *c*-Means (RFCM) [5] both used M-estimators in slightly different ways to cluster noisy datasets. The Fuzzy *c*-Least Median of Squares (FCLMS) is a robust clustering technique based on the LMS estimator [6]. Kim et al. [7] used the LTS estimator to build a prototype-based robust clustering algorithm known as the fuzzy trimmed *c*-prototypes (FTCP) algorithm. However, these attempts to use high breakdown estimators in a prototype-based clustering algorithm, by adjusting the FCM minimization functional, still suffer from FCM related problems. The goodness of the results is heavily contingent on a good initialization and none of the techniques guarantee an exact solution.

In this paper, we present a class of algorithms that implement the feasible solution algorithm for high breakdown estimators especially the LTS estimator which is guaranteed to lead to the exact LTS solution. The solution space is innovatively searched using a genetic algorithm with classical operators. In the next section, the feasible solution algorithm is described, followed by a discussion on the use of genetic algorithms and evolutionary techniques in data clustering in section III. In section IV, we propose the algorithm and in section V, we present experimental results, followed by conclusions in section VI.

II. The Feasible Solution Algorithm

The best LS regression fit is the line that minimizes the sum of squared residuals from the line to all the *n* random observations. If however, only *h* of these observations are true and the rest recorded erroneously, the best LS fit will be biased towards these erroneously-recorded leverage points. The LTS regression estimator rectifies this bias by minimizing the sum of squares of the smallest *h* of the total *n* residuals. The estimator achieves robustness by trimming the $(n - h)$ observations, the ones with the large residuals. The LTS functional is of the form,

Amit Banerjee is with the Evolutionary Computing Systems Laboratory, Department of Computer Science and Engineering, MS/171, University of Nevada, Reno, NV 89557, USA (email: banerjee@cse.unr.edu).

Sushil J. Louis is with the Evolutionary Computing Systems Laboratory, Department of Computer Science and Engineering, MS/171, University of Nevada, Reno, NV 89557, USA (phone: +1 775 784 4315; fax: +1 775 784 1877; email: sushil@cse.unr.edu).

$$\sum_{i=1}^{h}(r^2)_{i:n}$$

where $(r^2)_{1:n} \leq \ldots \leq (r^2)_{n:n}$ are ordered squared residuals.

The motivation for using the LTS criterion does not provide any clue to its implementation, which involves determination of which cases to trim. If the size of the dataset is small, then a combinatorial scheme which involves fitting regression to every possible subset of size h and subsequently finding the minimal residual sum of squares can be used. Also popular is the basic resampling scheme [2, 8], which uses elemental subsets of size p ($p << h$), where p is the dimension of the data set and hence, has manageable combinatorics. The fit obtained however, is crude with a high degree of approximation (depending on the number of elemental subsets used) and does not in general yield the exact LTS solution.

The exact LTS solution is the ordinary LS fit to some subset of size h of the data which cannot be improved by any single pairwise exchange of one observation in the subset for one out of the subset. This known form of the exact solution and the resulting necessary condition form the basis of the simulated annealing approach [9] and the feasible solution approach [10, 11]. These are probabilistic schemes with guaranteed convergence to a global optimum under certain assumptions. A feasible solution is defined as a local optimum fit obtained by a refinement (pairwise swapping) process from a randomly picked starting subset of size h, called the retained set. From a starting retained set, observations are swapped with the trimmed subset (of size $n - h$) and the starting retained set is modified to include the observation that produces the largest reduction in the residual sum of squares in the retained set. This modified starting subset is then subject to further refinement until no pairwise swap results in a reduction of the residual sum of squares. The retained subset is called a feasible solution and a single application of the refinement process will always lead to a feasible solution. To obtain the globally optimum fit, the refinement process needs to be repeated using distinct starting subsets and following each to its feasible solution. The exact solution is the feasible solution with the lowest residual of squares. For a detailed discussion of the theoretical aspects of the feasible solution algorithm, the reader is referred to [10, 11].

The feasible solution algorithm and its implementation provided the basis for the fuzzy least trimmed squares clustering (FLTS) algorithm [12], an unsupervised clustering methodology that uses a cluster validity index to pick an appropriate value of h. In the present work, the feasible solution algorithm is implemented as a structured search through the solution space, instead of a pairwise swap which is a computationally expensive process. The structured search is done using a genetic algorithm with classical genetic operators.

III. Genetic Algorithms In Clustering

Genetic algorithms (GA) are search algorithms based on the mechanics of natural selection and genetics [13]. A subclass of evolutionary computing-based search and optimization techniques, GAs differ from conventional search procedures in that they work with a coding of the parameter set, not the parameters themselves, and search for a solution from a population of prospective solutions (as against improving a single solution). The prospective solutions are first coded into binary chromosome strings, followed by an evaluation of each string to measure the goodness of the individual string. This is done by using a predefined problem-specific fitness function. As with nature, strings having higher fitnesses are preferred over others. These high-fitness individuals are selected to fill up a mating pool, a process known as *selection*. The individuals are then combined together using two genetic operators, called *crossover* and *mutation*, to produce offspring for the next generation. This process of selection, crossover and mutation is carried on iteratively until some predefined termination criterion is met. GAs distinguish themselves from other optimization techniques because of their implicit parallelism and diversity. In the last two decades, GAs have gained immense popularity as an optimization tool in engineering, social and physical sciences, computer sciences, and operations research among others.

Perhaps the first to use evolutionary techniques in fuzzy clustering were Bezdek et al. [14]; they performed a hard c-means taking the membership degrees and prototype locations as the parameters for the GA. Klawonn and Keller [15] argue that optimizing both the cluster prototypes and the membership matrix (a c x n real-valued parameter set) seems redundant, and as a result they optimize only the cluster prototypes. They achieved promising results partitioning datasets with solid rectangular clusters, and spherical clusters (typical for FCM), but not so optimistic results with shell-clustering. The genetically guided algorithm (GGA) was proposed to ameliorate FCM's overt dependence on initialization of prototypes [16]. A non-fuzzy clustering implementation using a GA-based optimization technique can also be found in [17], where instead of coding the parameters for the location of cluster prototypes, the investigators used a non-binary representation for their genotypes. In the resulting algorithm called the clustering genetic algorithm (CGA), the phenotypes are represented as one-dimensional integer arrays with $n + 1$ elements, the first n bits of the arrays represent cluster labels (1,.., c) for each datum and the last bit represents the number of clusters, c. The CGA, not only optimized the structure of the partition, it also made no *a priori* assumptions about the value of c. The only previous work in robust clustering using GAs is the hard (and fuzzy) implementation of LMS, called the C-LMedS (and Fuzzy C-LMedS) [6]. The c-prototypes to be optimized are coded and concatenated into a binary string, the fitness function is arbitrarily chosen as the fourth-root of the inverse of the LMS objective functional, and classical genetic operators are used (although there is very little information provided about the genetic selection procedure).

IV. The Genetic Fuzzy Least Trimmed Squares Clustering (Gen-FLTS) Algorithm

The family of feasible solution algorithms proposed by Hawkins [11] all note that the subset of retained entities giving rise to the exact optimum must satisfy the necessary condition stated as – the criterion cannot be improved by exchanging any of the currently trimmed entities for any of the currently retained entities. In other words, the retained set can be improved by populating it with good entities (and at the same time jettisoning noisy entities), until no further improvement is possible. GAs are tailor-made for such optimization problems. This section describes the GA used to optimize the construction of the retained set and the trimmed set.

A. Representation

Instead of parameterizing the *c*-prototypes, a binary representation similar to [17] is adopted. In our representation, the data are assigned either to the retained set or the trimmed set. The size of the trimmed set is not predefined – in other words, *h* is unknown. If *n* is the number of data points to be clustered, the 1s in the binary string of *n*-bits correspond to entities in the retained set, and the 0s correspond to entities in the trimmed set.

B. Fitness Evaluation

The strings produced in a particular population are evaluated relative to each other. The fitness function is the basis of this evaluation, and the objective of the GA is always to maximize the fitness of its strongest strings, thereby driving the process to a better solution. In our case, the retained set is partitioned by a fuzzy clustering routine (FCM) for a fixed value of *c*. The functional minimized by FCM is given by,

$$J_m = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^m \| x_k - v_i \|^2 \tag{1}$$

where u_{ik} is the membership of the datum *k* in cluster *i*, *m* is the fuzzifier (1.2 or 2.0 in our experiments), x_k is the datum representation of *k* in *d*-dimensions (a *d*-dimensional vector), v_i is the prototype representation of cluster *i* (in *d*-dimensions) and ||.|| is the inner product norm. However, the functional in (1) decreases as the size of the retained set gets smaller. In our case, since we make no assumptions on the amount of contamination present in the data (the size of the trimmed set is not known *a priori*), if the fitness function is considered to be some inverse function of (1), as in [6], then the algorithm will be driven to find the smallest (or the densest) regions of the data set, which may or may not correspond to an actual cluster. Hence, we need a function to compensate for the size of the retained set, of a form shown in (2),

$$J = J_m - h_n . a \tag{2}$$

where *a* is a suitably chosen scaling constant, and h_n is the size of the retained set ($h_n \le n$). The functional in (2) is then normalized based on the maximum value in the given population, so that the string with the maximum value of (2) gets a fitness of zero. In other words, we convert the ***min***(*J*) problem to a ***max***(*J*′) problem where,

$$J' = (J_{max} - J) / J_{max} \tag{3}$$

The string with the smallest *J* value thus gets to be the fittest string (fitness < 1.0). In case this smallest value in a given population is negative, we scale up all the *J*′ values by subtracting each of them by the smallest value.

C. Selection

The survival of the fittest ensures only the stronger individuals pass on their genetic material to the next generation. The average fitness of the new generation is higher than the fitness of the old one. In our experiments, the stochastic tournament selection technique [13] performed better than roulette wheel sampling, deterministic sampling, and remainder stochastic sampling. The sum of fitnesses of the individuals selected by the stochastic tournament selection was found to be consistently higher than that of the other selection procedures. An elitist strategy to retain the strongest individual for the next generation was later adopted in our experiments.

D. Crossover and Mutation

A single-point crossover operator combines two selected strings, at a randomly chosen crossover point. Depending on the probability of crossover, P_c, a maximum of *t*/2 crossover operations are possible, where *t* is the size of the population. Mutation is a bit-inversion operator carried out after crossover, with a predefined probability, P_m. The three operators – selection, crossover and mutation, generate a new population of size *t*, to replace the old population. Each string in the new population corresponds to a new retained set–trimmed set combination.

E. GEN-FLTS: The Algorithm

An initial population of size *t* is created with random *n*-bit binary strings. Data corresponding to unit locations in the string are then passed on to the clustering routine, which returns the functional value in (2). Strings are then evaluated on the basis of their fitness values in (3); a pool of size *t* is created that contains the fitter-than-usual strings, using either the stochastic tournament selection or the elitist tournament selection processes. A pair of strings from this pool is then mated (crossover and mutation, depending on P_c and P_m respectively), and a new population is created. As the fuzzy partition created by FCM depends heavily on the initialization procedure used, we perform ten FCM partitions every time and consider the average functional value over the ten runs. The process of selection, crossover, mutation, and creation is repeated for a fixed number of generations, usually a function of *t*.

V. Simulations

The proposed GEN-FLTS algorithm is tested on three different data sets. For the purpose of illustration, classification results on a simple two-cluster synthetic data set are first presented. The other data sets used are the birth-death data earlier analyzed in [12] and the Wisconsin breast cancer data, previously presented in [17].

A. *Example-1: Synthetic Two-Cluster Data*

This synthetic dataset shown in fig. 1 consists of two Gaussian clusters each comprised of 35 two-dimensional vectors with means and variance,

$$\mu_1=(10,15), \mu_2=(4,17.5), \Sigma_1=\Sigma_2=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

and corrupted by 30 uniformly distributed noise vectors. In other words, n = 100 and h = 70, although we assume no *a priori* knowledge of h.

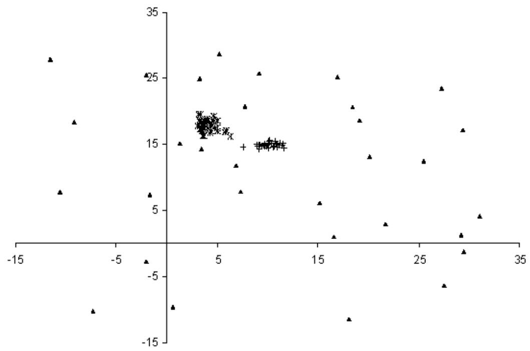

Fig. 1. Two-cluster data set (n = 100, h = 70, c = 2)

For all the simulations with this data set, we chose population size, t = n. The GEN-FLTS algorithm was run for a fixed number of generations (also equal to n). The data was so labeled that the first 70 bits of the string corresponded to the data in the two Gaussian clusters, and the last 30 positions were the noise datum. A visual examination of the best string after every generation was enough to confirm the efficacy of the proposed technique. The simulations were carried out for a range of combinations of P_c and P_m, and it was seen that the procedure quickly converges to the optimum solution for P_c = 0.9 and P_m = 0.01. However, choosing a correct value of a which compensates for reduction in the size of the retained set, was something we expected to learn from this simulation. We performed clustering for a = 1, 10, 50 and 100, and found that the solution only converges for a = 50 after about 50 generations (shown in fig. 2). If the whole dataset is clustered using FCM (n = 100, c = 2, m = 2.0), J_m = 5177.50 and J_m/n = 51.775 ~ 50. This provides us with a heuristic to chose the value of a, (this was later confirmed by the succeeding simulations). The maximum, minimum and the average values of J for P_c = 0.9 and P_m = 0.01, over 100 generations, at a = 50, are plotted in fig. 3. Because of random initialization of the population strings, we let the solution stabilize for the first 10 generations (the first 10 generations are hence omitted from the fig. 2 and 3).

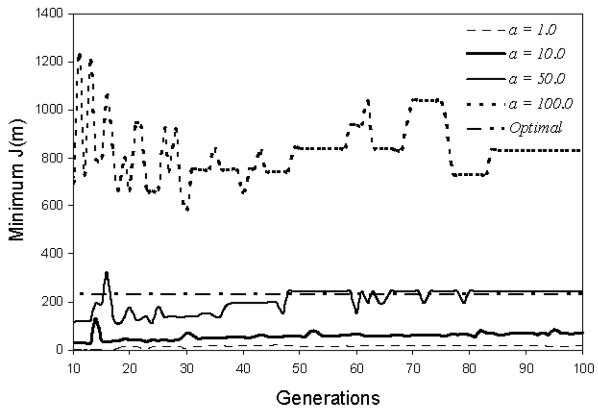

Fig. 2. Fittest Individual J_m vs. Generations for four values of a

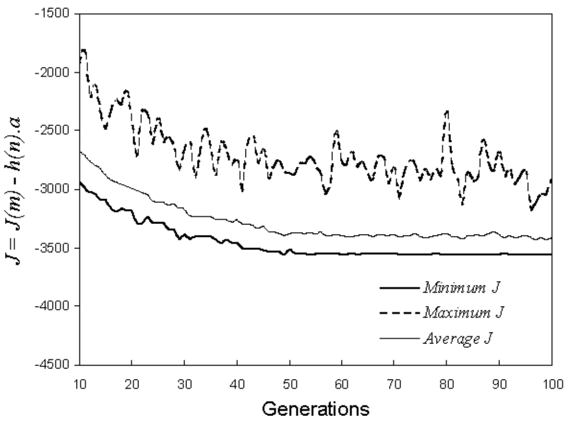

Fig. 3. Plot of Maximum, Minimum and Average value of the functional J vs. Generations for a = 50.0, P_c = 0.9, P_m = 0.01. (Note that smaller the value of J, better the fitness of the string)

B. *Example-2: Birth-Death Rate Data*

The birth-death data analyzed in [12] is revisited here. The data is a collection of birth and death rates of 70 countries and is shown as a scatter plot in fig. 4. The three outliers are Denmark (death rate exceeding birth rate), Ghana and Ivory Coast (abnormally high birth rates). Unlike the analysis in [12], we do not assume h = 67. The parameters for the algorithm are, a = 10, t = n and the algorithm is run for a maximum number of generations = 2*n (= 140). The strings are processed by an FCM routine (with m = 1.2) for c = 2 and c = 4. The two choices of c (which obviously correspond to either two clusters separated on the basis of birth rates, or further classified on their death rates into four clusters) does not seem to affect the convergence of the strings and the simulation converges very quickly sometimes within 20 generations. When the noisy data set (n = 70) is clustered with FCM, we get J_m = 3180.33 and J_m/n = 45.5; we hence tried using a = 50 for this set of simulations as well. Selecting a value for a based on J_m/n seems sound because a = 50 is the only test case that converged to the exact solution (correctly identified the three outliers).

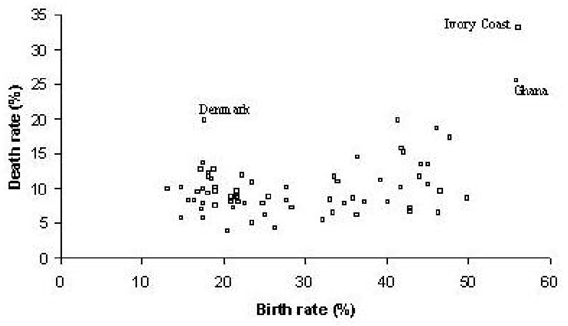

Fig. 4. The birth-death rate data set ($n = 70$, $h = 67$, $c = 2$ or 4)

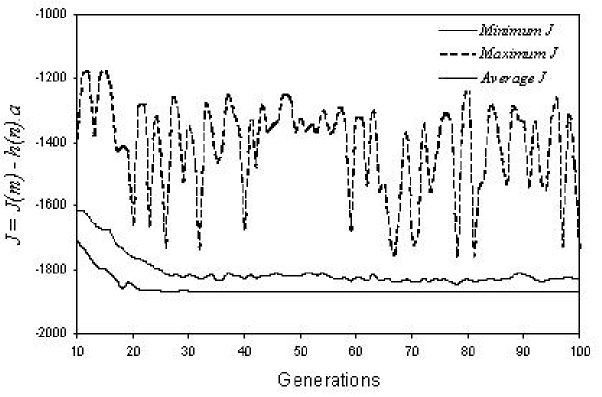

Fig. 5. Plot of Maximum, Minimum and Average value of the functional J vs. Generations for $a = 50$, $P_c = 0.9$, $P_m = 0.001$, $c = 2$

C. *Example-3: Wisconsin Breast Cancer Data*

This database in available at the UCI Machine Learning Repository [18] and has been analyzed as a two-class problem in [17]. Each entity is defined by nine attributes (ranks in the range of 1-10) and an associated class label (benign or malignant). The two classes are known to be linearly inseparable. Hence, the smaller class (malignant data) can safely be assumed to be noise among datum corresponding to the larger class (benign entities). There are 699 total cases out of which 16 have a single missing feature. These cases were removed and the remaining 683 cases (444 benign and 239 malignant – 35% noise) in nine dimensions constituted the input data for this problem. Chakravarthy and Ghosh [19] report uncovering three significant clusters using radial basis functions for classification. Kothari and Pitts [20] found two clusters with a 94.28 % accuracy of classification. This was followed by the CGA [17] reporting 95.75% accuracy in finding the two classes. In this work, we use a range of cluster values c, to test the classification accuracy in detecting the 239 malignant cases (classified as noise). The best results are obtained for $c = 5$, with 236 out of 239 malignant cases uncovered which translated to an accuracy of 98.75%. This however, seems to be heavily dependent on the choice of (1) P_c and P_m, (2) the selection criteria, and (3) the number of clusters uncovered in the benign set, c (retained set), but not so much on the choice of a (as long as $a \sim J_m/n$, = 10 in this case). The classification accuracy falls for values of $P_c <$ 0.8 and $P_m > 0.01$. It is also seen that for a population size of $t = 300$ (maximum number of generations allowed = 300), the stochastic tournament selection scheme fails to retain the fittest string across generations, forcing us to use an elitist strategy that preserves the fittest individual. Another tournament selection strategy that preserves half (or less than half) of a generation is also found to work well although the implementation is considerably slower.

VI. Conclusions

The fuzzy-LTS (FLTS) clustering scheme, built on the high breakdown LTS estimator, is known to be extremely robust, although the feasible solution algorithm is one of the only two known implementations of the LTS estimator that guarantees a global solution. The FLTS feasible solution algorithm implementation in [12] is computationally expensive. The evaluation of the trimmed and retained set (by pairwise swapping of entities between the sets) is a classic optimization problem, and is perfect fodder for a genetic algorithm. An elegant implementation of the FLTS feasible solution algorithm using a genetic algorithm for optimization has been presented in this paper, and results of clustering on three datasets of varying degrees of complexity have been presented. These results have been compared to those reported on the same datasets in literature and have been shown to be superior in many cases.

Although quite promising, there are a lot of avenues open for improvement. A better fitness functional needs to be investigated, a function perhaps on the lines of $s(i)$ proposed in [17]. The sensitivity of the value of a on the overall fitness of an individual also needs to be studied. Lastly, as an addendum to the algorithm presented in this paper, we are working on a real-time robust clustering procedure to simultaneously ascertain the correct number of the number of clusters.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant no. 0447416.

References

[1] A. F. Siegel, “Robust regression using repeated medians,” *Biometrika*, 69, pp. 242-244, 1982.

[2] P. J. Rousseeuw, “Least median of squares regression,” *Journal of the American Statistical Association*, 79, pp. 871-880, 1984.

[3] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, New York: Wiley, 1987.

[4] H. Frigui and R. Krishnapuram, “A robust clustering algorithm based on M-estimator,” *Proc. First International Conference on Neural, Parallel and Scientific Computations*, Atlanta, GA, May 1995, pp. 163-166.

[5] Y. Choi and R. Krishnapuram, “Fuzzy and robust formulations of maximum-likelihood-based Gaussian mixture decomposition,” *Proc. IEEE International Conference on Fuzzy Systems*, New Orleans, LA, Sept. 1996, pp. 1899-1905.

[6] O. Nasraoui and R. Krishnapuram, “A genetic algorithm for robust clustering based on a fuzzy least median of squares criterion,” *Proc. Annual Meeting of the North American Fuzzy Information Processing Society*, Syracuse, NY, Sept. 1997, pp. 217-221.

[7] J. Kim, R. Krishnapuram and R. N. Dave, “Application of the least trimmed squares technique to prototype-based clustering,” *Pattern Recognition*, vol. 17, pp. 633-641, 1996.

[8] A. Marazzi, “Algorithms and programs for robust linear regression,” in *Directions in Robust Statistics and Diagnostics: Part I*, Edited by W. Stahel and S. Weisberg, pp. 183-199, New York, NY: Springer Verlag, 1991.

[9] A. C. Atkinson and S. Weisberg, “Simulated annealing for the detection of multiple outliers,” in *Directions in Robust Statistics and Diagnostics: Part I*, Edited by W. Stahel and S. Weisberg, pp. 7-20, New York, NY: Springer Verlag, 1991.

[10] D. M. Hawkins, “The feasible solution algorithm for least trimmed squares clustering,” *Computational Statistics and Data Analysis*, vol. 17, pp. 185-196, 1994.

[11] D. M. Hawkins and D. J. Olive, “Improved feasible solution algorithms for high breakdown estimation,” *Computational Statistics and Data Analysis*, vol. 30, pp. 1-11, 1999.

[12] A. Banerjee and R. N. Dave, “The feasible solution algorithm for fuzzy least trimmed squares clustering,” *Proc. Annual Meeting of the North American Fuzzy Information Processing Society*, Banff, Canada, June 2004, pp. 222-227.

[13] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Boston: Addison-Wesley, 1989.

[14] J. C. Bezdek, S. Boggavarapu, L. O. Hall and A. Bensaid, “Genetic algorithm guided clustering,” *Proc. First IEEE Conference on Evolutionary Computation*, Orlando, FL, June 1994, pp. 34-39.

[15] F. Klawonn and A. Keller, “Fuzzy clustering with evolutionary algorithms,” *International Journal of Intelligent Systems*, vol. 13, pp. 975-991, 1998.

[16] L. O. Hall, I. B. Ozyurt and J. C. Bezdek, “Clustering with a genetically optimized approach,” *IEEE Transactions on Evolutionary Computation*, vol. 3(2), pp. 103-112, 1999.

[17] E. R. Hruschka and N. F. F. Ebecken, “A genetic algorithm for cluster analysis,” *Intelligent Data Analysis*, vol. 7, pp. 15-25, 2003.

[18] C. J. Merz and P. M. Murphy, UCI Repository of Machine Learning Databases, [http://www.ics.uci.edu/~mlearn]. University of California, Irvine, CA.

[19] S. V. Chakravarthy and J. Ghosh, “Scale-based clustering using radial basis function network,” *IEEE Transactions on Neural Networks*, vol. 7(5), pp. 1250-1261, 1996.

[20] R. Kothari and D. Pitts, “On finding the number of clusters,” *Pattern Recognition Letters*, vol. 20, pp. 405-416, 1999.