Automatic Shell Clustering Using a Metaheuristic Approach

Siddharth Pal[1], Anniruddha Basak[1], Swagatam Das[1],
Ajith Abraham[2] and Vaclav Snasel[3]
[1]Department of Electronics and Telecommunication Engineering, Jadavpur University, India
[2] Machine Intelligence Resarch Labs (MIR Labs), USA. Email: ajith.abraham@ieee.org
[3]VSB Technical University Ostrava, Czech Republic. Email: vaclav.snasel@vsb.cz

***Abstract*—This paper proposes a simple, metaheuristic clustering technique, inspired by the mountain clustering method of Yager and Filev, for detecting general quadric shell type clusters. The algorithm employs an ecologically inspired metaheurisitc algorithm, called Invasive Weed Optimization (IWO) to evolve a set of cluster prototypes in the shape of curves/hyper-surfaces. The objective function is modeled using the concept of the mountain function from Yager and Filev's work. The metaheuristic approach can be extended to solid clusters and various shell clusters like circular, elliptical, rectangular etc. The proposed method is tested on several synthetic datasets as well as real images to detect circular and elliptical shell clusters and the results obtained are found to be very promising.**

***Keywords* – shell clustering, invasive weed optimization, mountain and subtractive clustering, circle detection, shape recognition.**

I. Introduction

Traditional fuzzy clustering algorithms like Fuzzy C – Means (FCM) [1] and Possibilistic C – Means (PCM) [2] cannot detect clusters that lie in nonlinear subspaces of the feature space because they use points (i.e. cluster centroids) as prototypes. To find clusters in nonlinear subspaces that resemble *shells* or patches of hyper-surfaces with no interior points, prototypes like curves/hyper-surfaces have been proposed. The shell clustering techniques provide an effective means for solving the problem of fitting multiple curves/hyper-surfaces to unlabeled, sparse, and scattered data. Algorithms dedicated to detect shell type clusters have also found applications in boundary detection, surface approximation and similar computer-vision tasks [3 - 5]. A few representative fuzzy shell clustering algorithms are Adaptive Fuzzy C Shells (AFCS) algorithm [6], Fuzzy C Quadric Shells (FCQS) and its variants [7, 8], Fuzzy C Plano-Quadric Shells [9] etc. and they attempt to minimize the weighted squared sum of distances of a feature point to a prototype by updating the fuzzy membership and parameters in an alternating fashion. Most of the shell-clustering algorithms available in the literature are computationally quite expensive since either they need to perform matrix inversions or they solve some nonlinear equations iteratively. Usually for better results a series of algorithms need to be applied on the data.

Yager and Filev proposed the Mountain Clustering Method (MCM) [10], to estimate the cluster prototypes in a simple way. MCM provides an approximate estimation of the cluster centroids by constructing and destroying a mountain function on the grid space. It is suitable for applications where only an approximate set of cluster centroids will serve the purpose. Nevertheless, the cluster centroids found by MCM can be refined by other complicated and more involved clustering algorithms, when accuracy becomes the major requirement. MCM is less sensitive to noise than other competing clustering algorithms such as FCM [11]. To reduce the computational complexity of mountain clustering, Chiu proposed a slightly different variant known as the Subtractive Clustering Method (SCM), where calculating the mountain function is done on the data points rather than on the grid points [12].

Pal and Chakraborty [11] extended MCM for detecting circular shell-shaped clusters and proposed the Mountain Circular Shell (MCS) method. In this article we propose a metaheuristic shell clustering method that exploits the idea of mountain functions to evaluate the closeness of the data-points to the cluster prototypes and employs a powerful global optimization algorithm of current interest, called Invasive Weed Optimization (IWO) [13], to refine those prototypes. Our method is capable of detecting general quadric shell-shaped clusters and differs significantly from [11].

Pal and Chakraborty [11] considered circular shells where a shell prototype consists of two elements – centre of circle and radius of circle. The mountain function was calculated for various combinations of centres and radii. Then the derivative-based steepest ascent algorithm was applied to tune the obtained prototypes. After a circular shell was detected the mountain function was updated to negate the effect of previously detected shell prototypes. In this work, we do not divide the feature space into grids. Rather we evolve the shell cluster prototypes to fit them to the actual data-points by a metaheuristic algorithm. Our objective function that measures how much the distribution of the data-points deviate from the actual cluster prototype (which is a curve/hyper-surface) is based on the mountain function, which gives us the advantage of immunity to noise.

Our approach is free from the adjustments of the quantization step-size and other shortcomings of the steepest ascent algorithm (like trapping in a local optimum). The parameters of the cluster prototype are taken as search variables. In MCS method, the mountain function update strategy sometimes produces erroneous results where a previously detected prototype is detected once again because of imperfect discounting. We found this to happen when two cluster prototypes were quite close and one prototype contained significantly more number of points than the other. Thus when we are detecting a cluster prototype, we completely remove the corresponding data-points rather than update the mountain function. This produced a significant improvement in the clustering efficiency.

We begin with a brief description of the mountain and subtractive clustering methods in Section 2. Next we outline the classical IWO and its modifications proposed by us for enhancing the optimization performance. The proposed shell clustering technique is presented in Section 4. We provide the experimental results on several synthetic datasets and real-life images in Section 5 and finally the paper is concluded in Section 6 with some discussions on the future research issues.

II. Mountain and Subtractive Clustering

In MCM we discretize the *p*-dimensional bounded hyperspace into *p*-dimensional grids. This results in grid-points. The grid-points so produced are treated as candidate cluster prototypes. Once a cluster centre is estimated, the mountain function is updated to eliminate the effects of the already detected centres. The Mountain function is defined for each grid-point $\vec{v}_i$ as:

$$M(\vec{v}_i) = \sum_{k=1}^{n} e^{-\alpha . d(\vec{x}_k, \vec{v}_i)} \quad (1)$$

where α is a positive constant, $d(\vec{x}_k, \vec{v}_i)$ denotes the distance of $\vec{x}_k$ from the grid-point $\vec{v}_i$ .The mountain function can be viewed as a measure of the density of data-points in the neighbourhood of a grid-point. The mountain function in (1) had been defined to find centres of solid clusters. Yager and Filev suggested the update equation for the mountain function as follows

$$M_k(\vec{v}_i) = M_{k-1}(\vec{v}_i) - M^*_{k-1} e^{-\beta . d(\vec{v}^*_{k-1}, \vec{v}_i)} \quad (2)$$

In the abobe equation $M_k(\vec{v}_i)$ is the new mountain function, $M_{k-1}(\vec{v}_i)$ is the previous mountain function and $\vec{v}^*_{k-1}$ is the last detected center, and β is a positive constant. Using the discounted function given in equation (2) new cluster centres are detected until the level of the current maximum M^*_{k-1} falls below a certain level compared to the original maximum M^*_1. The process of finding new cluster centres is terminated when

$$\frac{M^*_{k-1}}{M^*_1} < \delta \quad (3)$$

where δ is a positive constant less than 1. Thus the parameters of the algorithm are α , β and δ .

The subtractive clustering technique is a variation of the mountain method which is computationally less expensive. Unlike the mountain method here each data-point is treated as a potential cluster centre. This method also takes the help of mountain function and subsequently discounts the mountain function, but does not use the concept of grids. Here only the data-points are tested as prospective cluster centres. Thus complexity of algorithm does not depend on the dimensionality or the spread of the data but on the number of data-points. The inherent problem with this method is that it will give good results only if the desired cluster centres is close to one of the data-points.

III. Invasive Weed Optimization (IWO)

In recent past, the computational cost having been reduced almost dramatically, researchers all over the world are paying a considerable amount of attention towards bio-inspiration and bio-mimicry, for solving computational problems and constructing intelligent systems like autonomous robots. Following this tradition, in 2006, Mehrabian and Lucas proposed the Invasive Weed Optimization (IWO) [13], a derivative-free, metaheuristic algorithm, mimicking the ecological behavior of colonizing weeds. Since its inception, IWO has found successful applications in many practical optimization problems like optimization and tuning of a robust controller [13], optimal positioning of piezoelectric actuators [14], developing a recommender system [15], antenna configuration optimization [16], design of E-shaped MIMO Antenna [17], design of encoding sequences for DNA computing [18], and design of compact U-array MIMO antenna design [19].

The basic idea of IWO goes like this: flowering plants reproduce seeds, which spatially disperse over a certain area and grow to plants. When the population of the plants is excess to fit in that area, only the plants with better fitness can survive. This process continues generation after generation. Main steps of the algorithm have been summarized below.

1. Initialization:

A finite number of weeds are initialized randomly in the *d*-dimensional search space

2. Reproduction:

Each member of the population is allowed to produce seeds depending on its own, as well as the colony's lowest and highest fitness, such that, the number of seeds produced by a weed increases linearly from lowest possible seed for a weed with worst fitness to the maximum number of seeds for a plant with best fitness. The number of seeds generated by a plant is as follows:

$$seeds = \max\left[floor\left\{ \max\_seed \times \left(\frac{worst^{fitness} - plant^{fitness}}{worst^{fitness} - best^{fitness}} \right) \right\}, 1 \right] \tag{4}$$

3. Spatial distribution:

The generated seeds are randomly distributed on the search space according to normal distribution with zero mean and normalized standard deviation σ_t . This is determined by the following equation:

$$\sigma_t = \left(1 - \frac{t}{iter_{\max}}\right)^n (\sigma_{initial} - \sigma_{final}) + \sigma_{final} , \tag{5}$$

where $iter_{\max}$ is maximum iterations, t is current iteration and n is the nonlinear modulation index. In the optimizing functions considered, the bounds differ from dimension to dimension. So we have introduced a concept of normalized standard deviation. This step ensures that the produced seeds will be generated around the parent weed, leading to a local search around each plant. However, the standard deviation σ_t of the random function is made to decrease over the iterations. We associate a standard deviation for each dimension which in itself is dependant on σ_t .

$$\sigma_{i,t} = c \times \sigma_t \times (UB_i - LB_i) \tag{6}$$

$\sigma_{i,t}$ is the standard deviation pertaining to the i^{th} dimension. UB_i and LB_i are the upper and lower bounds for the i^{th} dimension. c is chosen as 0.25. This step ensures that the probability of dropping a seed in a distant area decreases nonlinearly with iterations, which results in grouping fitter plants and elimination of inappropriate plants

4. Competitive Exclusion:

There is a need of some kind of competition between plants for limiting maximum number of plants in a colony. Initially, the plants in a colony will reproduce fast and all the produced plants will be included in the colony, until the number of plants in the colony reaches a maximum value $pop_{\max}$. However, it is expected that by this time the fitter plants have reproduced more than undesirable plants. From then on, only the fittest plants, among the existing ones and the reproduced ones; are taken in the colony and the steps 1 to 4 are repeated until the maximum number of iterations has been reached.

In order to improve the performance of the classical IWO, in this article we have modified (5) as:

$$\sigma_t = \left(1 - \frac{t}{iter_{\max}}\right)^n |\cos(t)| (\sigma_{initial} - \sigma_{final}) + \sigma_{final} \tag{7}$$

The $|\cos(t)|$ term adds a variation in standard deviation, which helps in exploring the better solutions quickly and prevents the new solutions from discarding an optimal solution when σ_t is relatively large. Suppose we consider an optimization problem f(x) which needs to be minimized. In classical IWO the seeds are generated from a plant with a certain standard deviation σ_t , which is decreased as number of iteration increases. Thus the plants slowly undergo a behavioral transformation from an explorative nature to an exploitative one. This modification is proposed, such that if the weeds are near an optimal solution then it can exploit it quickly rather than wait for the standard deviation to decrease to a reasonable value which might be achieved near the end of the run. In our proposed strategy the standard deviation actually varies within an envelope, so lesser values of σ_t are obtained much before the end of the run. This facilitates quicker detection of optimal solutions and better results as compared to classical IWO.

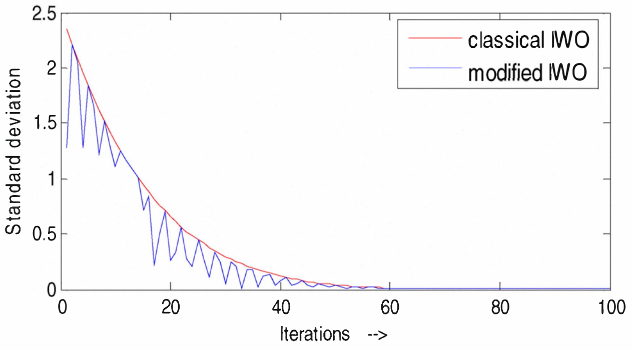


Figure 1. Comparison of the variations of standard deviation with iterations for the classical and modified IWO

IV. The Proposed Shell Clustering Algorithm

In this Section, we outline the proposed shell clustering algorithm based on the concepts of the mountain function and modified IWO. Suppose each data-point is described by an *n*-dimensional feature vector like $\vec{x} = [x_1, x_2, ... x_n]^T$.

The prototypes consist of parameter vectors $\vec{A}_i$ that define the equations of the hyperquadric surface. Once the parameter vector is determined we can detect whether a certain datapoint lies on the surface or not. The general equation of a hyperquadric surface is given as

$$\vec{A}_i^T \vec{B} = 0 \tag{8}$$

where,

$\vec{A}_i^T = [a_{i1}, a_{i2}, ...a_{in}, a_{i(n+1)}, ...a_{ip}]$

$\vec{B} = [x_1^2, x_2^2, ...x_n^2, x_1 x_2, ...x_{n-1} x_n, x_1, ...x_n, 1]$

$a_{i1} \to a_{in}$ represents the coefficients of x_i^2

$a_{i(n+1)} \to a_{i(n^2/2+n/2)}$ represents the coefficients of $x_i x_j$

$a_{i(n^2/2+n/2+1)} \to a_{i(n^2/2+3n/2)}$ represents the coefficients of x_i

a_{ip} represents the constant term coefficient with $p = n^2/2 + 3n/2 + 1$.

Since our objective is to fit the hyperquadric surfaces to the data-points, we encode the parameters of a number of such candidate hyperquadric surfaces as the population members or search-agents which are initialised randomly and then gradually evolved with the modified IWO to fit to the actual distribution of the data-points. The mountain function for the *i*-th hyperquadric surface is defined as:

$$M(\vec{A}_i) = \sum_{k=1}^{n} e^{-\alpha\left|\vec{A}_i^T.\vec{B}_k\right|} \tag{9}$$

Here *n* is the number of data-points in the set. The quantity $\left|\vec{A}_i^T.\vec{B}_k\right|$ will be minimized for the k^{th} data-point when it lies on the hyperquadric surface defined by parameter vector $\vec{A}_i$. Thus the contribution of the k^{th} data-point to the mountain function will be close to 1 if it lies on the candidate hyperquadric surface. If the parameter vector of the i^{th} agent $\vec{A}_i$ represent a hyperquadric surface present in the dataset then the mountain function will be maximized for that agent. Thus, the objective function to be minimized for the i-th search agent is given as:

$$f_i = -M(A_i) \tag{10}$$

For detecting circular shells, the hyperquadric surface can be simplified to:

$$f_i = -\sum_{k=1}^{n} e^{-\alpha\left|(x_1-a)^2+(x_2-b)^2-R\right|} \tag{11}$$

In this case a population member of IWO will contain the centre and radius of the candidate circular shell. Once a parameter vector with best fitted circular shell is found, we classify the data-points depending whether they fall in the present circular shell or not.

If $e^{-\alpha\left|\vec{A}_i^T.\vec{B}_k\right|} \geq \rho$ then we say that the k^{th} data-point lies on the circular shell described by $\vec{A}_i$. We have chosen $\rho = 0.95$ for all the datasets that we have considered. Suppose the present set of data-points is denoted by Ψ. Let us define a set,

$$\zeta = \left\{k \in \Psi \middle| e^{-\alpha\left|\vec{A}_i^T.\vec{B}_k\right|} \geq \rho\right\} \tag{12}$$

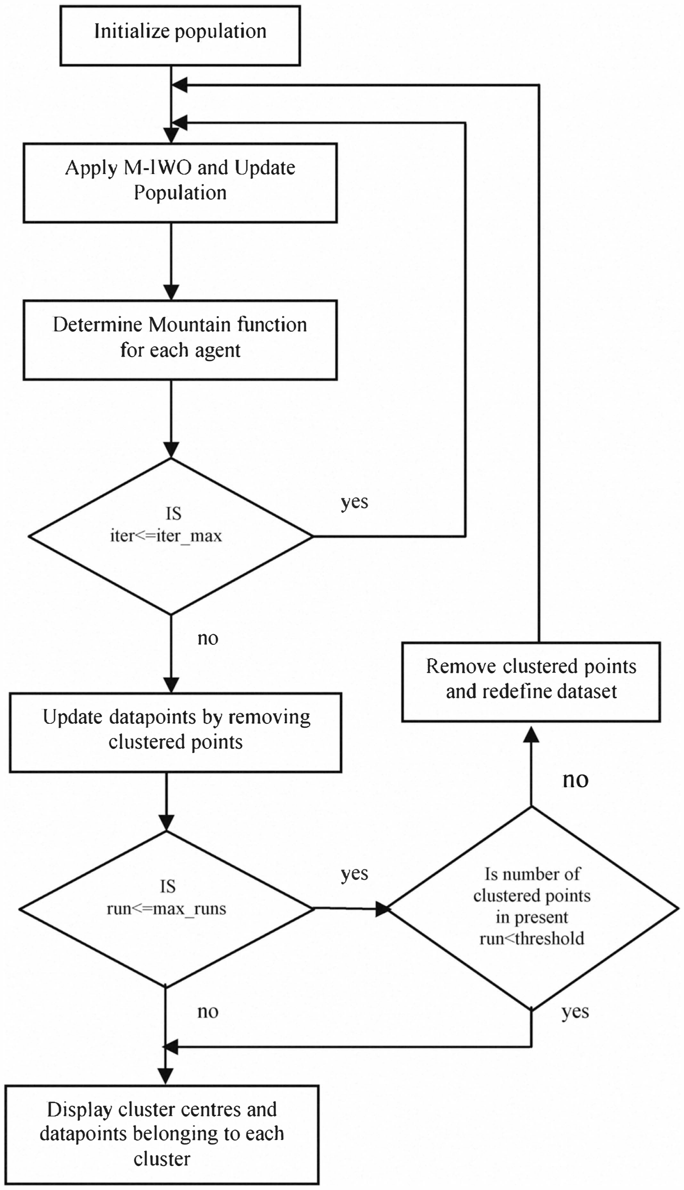

Initialize population
Apply M-IWO and Update Population
Determine Mountain function for each agent
IS iter<=iter_max
yes
no
Update datapoints by removing clustered points
IS run<=max_runs
yes
no
Is number of clustered points in present run<threshold
no
yes
Remove clustered points and redefine dataset
Display cluster centres and datapoints belonging to each cluster

Figure 2. Flowchart illustrating the Automatic Shell Clustering Algorithm

The set of data-points ζ are said to be clustered in a shell described by parameter vector $\vec{A}_i$. These clustered points are removed from the original set. The new set of data-points is defined as,

$$\Psi' = \{k \in \Psi | k \notin \zeta\} \tag{13}$$

To facilitate automatic detection we need to have a terminating condition for the optimization algorithm. The terminating condition occurs when number of runs is equal to *max_runs*. We also want the clustering process to terminate when all the shell clusters have been detected. We can introduce a terminating condition as in equation 10.

$$n(\xi) < \mathrm{N}_{\min}, \tag{14}$$

where

$n(\,)$ denotes the cardinal number of the set

$\mathrm{N}_{\min}$ is the minimum number of points required to be in a shell.

For our purpose we have chosen $\mathrm{N}_{\min}$ as $n(\Psi)/20$ .Here Ψ denotes the set of all data-points.

Figure 2 shows flow-chart of the complete clustering process.

In previous works on mountain or subtractive clustering the mountain function was either evaluated for all the datapoints or the parameter hyperspace was divided into grids and the mountain function was calculated for the grid points. In [8] the refinement of cluster centres was carried out by a steepest ascent method. Evidently in this article we differ considerably from such approach.

Moreover most of the works on subtractive clustering have used a discount function to update the mountain function. However in our approach after a certain shell has been detected we change the dataset by removing the clustered points rather than updating the mountain function. Merely updating the mountain function and keeping the original set of points leads to a fitness function which doesn't yield correct results. This is because exact discounting can never take place. We cannot nullify the effect of a neighbouring cluster completely. If a cluster has been detected with many points and in the next iteration we are supposed to detect a cluster with far less points then it may happen that the previous cluster is once again detected because the mountain function value corresponding to the earlier shell still dominates the smaller shell even after discounting. Even with a judicious choice of discounting function the problem hyperspace because more complicated and the efficiency of the overall algorithm tend to decrease. Thus for regular shell cluster we do not need to use the method of discounting. We simply remove the previous detected clusters. However while dealing with solid clusters we need to take help of the method of discounting.

V. Experimental Results

The proposed algorithm is tested with seven synthetic datasets and two images with varying degrees of complexity. On the images, the primary objective was to detect the circular boundaries automatically, which stand as an important problem in vision. Characteristics of the seven synthetic datasets have been listed in Table 1.

Table 1: Description of Datasets

Dataset No.	Comments
1	Two shell clusters overlap. This does not cause any problem to the algorithm as in the parameter space they are widely separated.
2	Here the two circular shells are concentric.
3	This dataset contains two semicircular shells. This tests the algorithm if it can detect incomplete circular shells.
4	This dataset consists of two circular shells and a semicircular shell inside a large circular shell.
5	This dataset consists of 8 circular shells. It tests the algorithm's capacity to correctly detect multiple intersecting shells.
6	This dataset consists of two thick semi-elliptical shells.
7	This dataset contains two intersecting ellipses.

We considered two-dimensional datasets consisting of elliptical and spherical shell clusters for the ease of visualization. The modified IWO-based proposed algorithm was run with the following parametric setup - maximum number of iterations $iter_{\max}$ = 60, initial normalized standard deviation $\sigma_{initial} = 1$, final normalized standard deviation $\sigma_{final} = .001$,maximum number of seeds *max_seeds*=5 and maximum number of plants $pop_{\max} = 100$ .The upper and lower bounds UB_i and LB_i were determined from the ranges of the dataset. The various parameters pertaining to the mountain clustering module were set as - $\alpha = 10, \rho = 0.95$. However for thick clusters such as in dataset 6 a low value of $\rho = 0.2$ had to be chosen

We present the final result in the form of an image which clearly depicts the clustered shells. For the sake of space economy in Figures 4 to 10 we show the clustered datasets 1 to 7 as obtained with the proposed algorithm. Since IWO is a stochastic optimization algorithm, results of two repeated runs on the same problem may not match completely. Hence we took 25 independent runs of the proposed algorithm on each of the datasets. In table 2 we provide the number of successful runs for the proposed algorithm and the clustering efficiency, defined as the percentage of successful runs over each problem. We also provide the average CPU time (in seconds) taken by the algorithm per run on each dataset.

A trial is said to be feasible if all the shell clusters were correctly identified. A trial is said to be successful if all the data-points were correctly clustered. Till now we have tested our algorithm on various datasets with varying degree of complexity. Here we would also like to apply our algorithm on various images for circular and elliptical shape detection. Prior to applying the evolutionary shell clustering technique, we need to convert the image into a binary edge image.

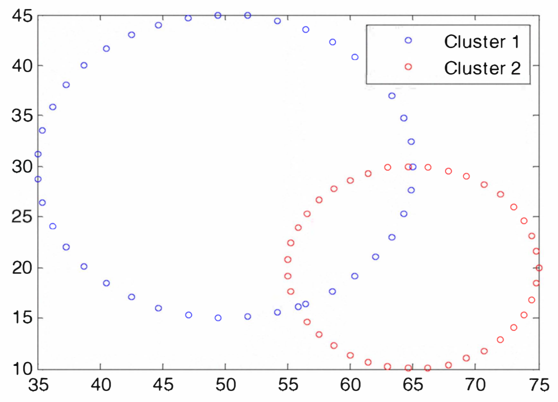


Figure 4: Clustered Result(Dataset 1)

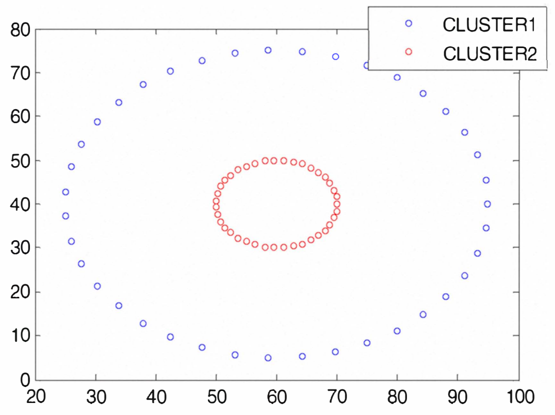


Figure 5: Clustered Result(Dataset 2)

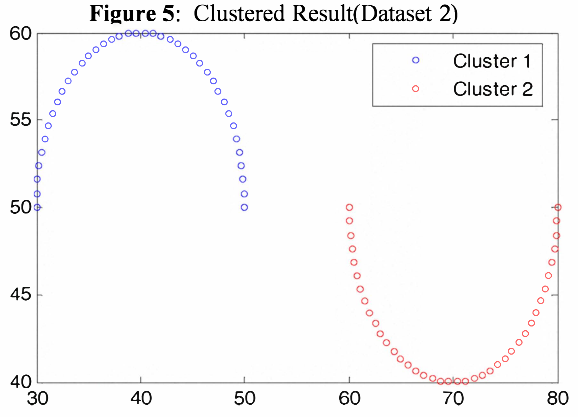


Figure 6: Clustered Result (Dataset 3)

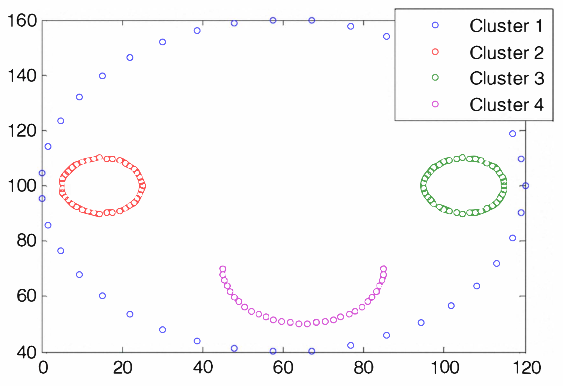


Figure 7: Clustered Result(Dataset 4)

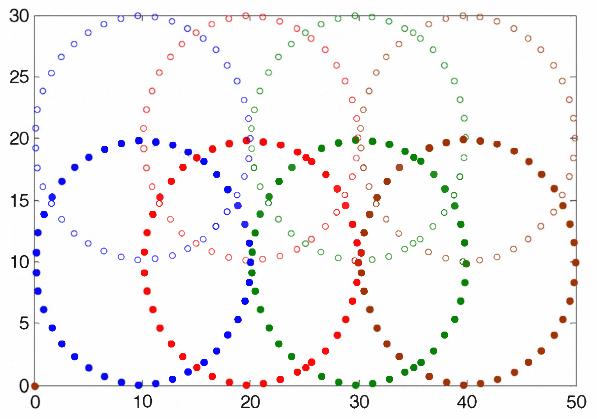


Figure 8: Clustered Result (Dataset 5)

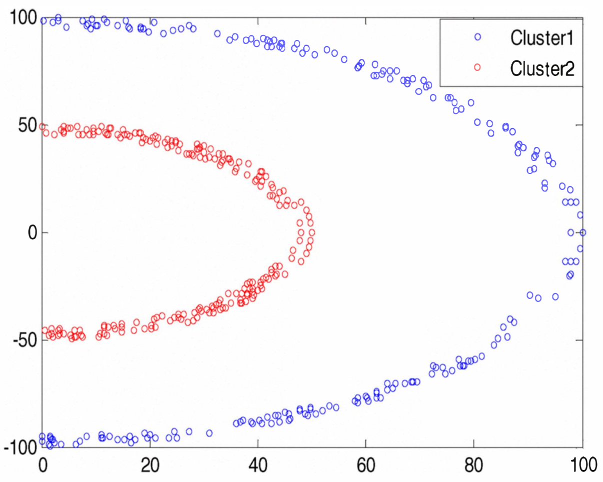


Figure 9: Clustered Result (Dataset 6)

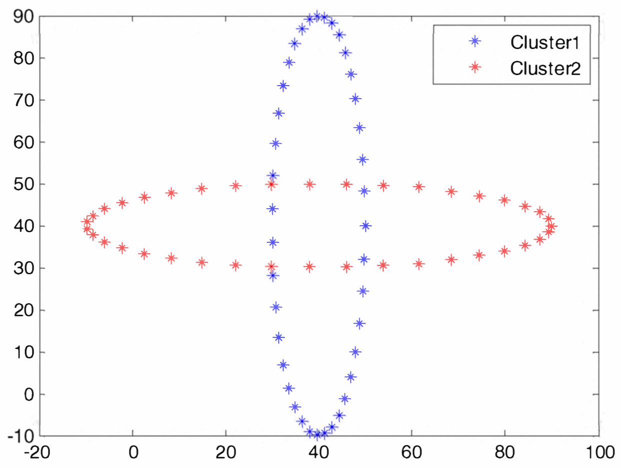


Figure 10: Clustered Result (Dataset 7)

The images shown in Figures 11 and 12 were first converted to binary images by simple thresholding function using MATLAB. The 1st Test image has a circle and other line segments intersecting the circle in various regions. The algorithm detects the circle correctly in all the runs as shown in Figure 11. In the 2nd test image we have a similar image but with added salt and pepper noise.

Table 2: Statistical Results

Dataset No.	No. of feasible runs	No. of successful runs	Clustering Efficiency	Avg. C.P.U Time per run (in Seconds)
1	25	25	100%	15.24
2	25	25	100%	12.12
3	25	25	100%	18.48
4	25	25	100%	28.40
5	25	25	100%	30.20
6	22	21	84%	17.72
7	25	23	92%	15.80

The added noise doesn't hamper the performance of the algorithm in any fashion. This is by virtue of the mountain function that we are using which is quite immune to added noise. The detected image is shown in Figure 12.

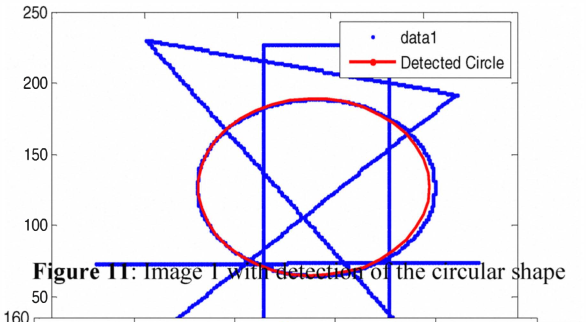


Figure 11: Image 1 with detection of the circular shape

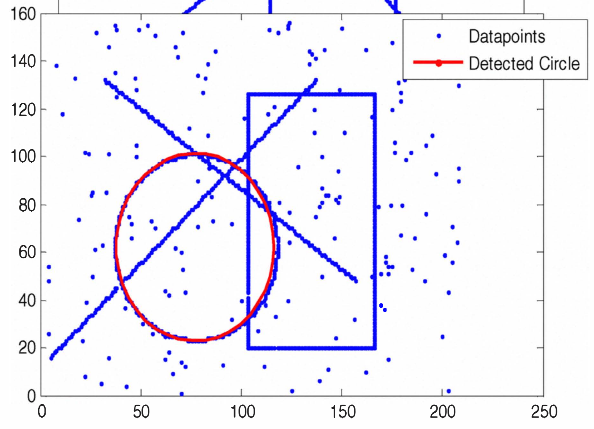
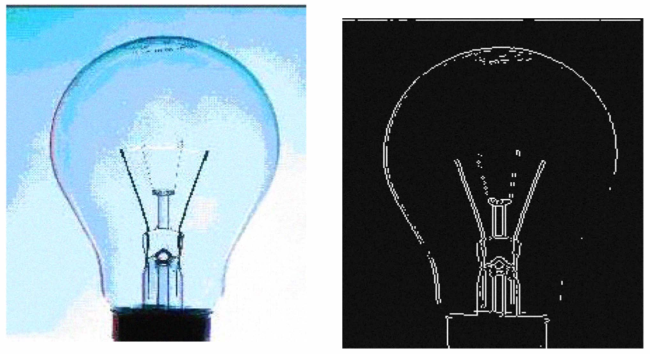
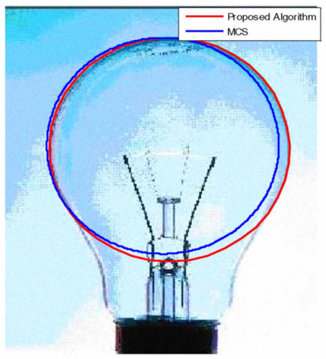


Figure 12: Image 2 with detection of the circular shape

Next we test our algorithm on two more real world images featuring an electric bulb and two coins. The objective is to identify the circular boundaries of the real-life objects from the edge image, which in these two cases were generated by using the Sobel edge-detector [20]. In Figures 13 (a), (b), and (c) we show the actual bulb image, its corresponding edge image and the detected circular boundaries by our proposed algorithm and the MCS method as obtained from [11]. For MCS we used the best parametric setup as specified by the authors in [11]. Figures 14 (a) – (c) exactly correspond to Figures 13 (a) – (c) but for the coin image.



(a) Image of a bulb (b) Sobel Edge Image of (a)

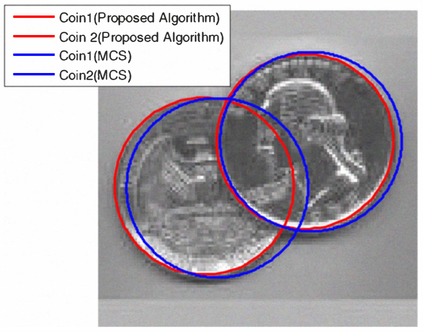


(c) Image showing detection of circular part of a bulb

Fig ure 13: Testing the proposed algorithm on a bulb image

(a) Image of two coins (b) Sobel Edge Image of (a)

Our simulation results indicate that the proposed modified IWO based shell clustering technique can efficiently detect circular and spherical shell clusters from a given dataset and it can correctly detect the number of clusters on the run. As is evident from the high clustering efficiency, the algorithm is fairly robust against different orientations of the shells (separable, non-separable, intersecting etc.) and also to the presence of noise, as can be perceived from Figure 12.



(c) Image showing detection of the circular boundaries of the coins

Figure 14: Testing the proposed algorithm on the image of two coins

Figures 13 and 14 and also the complete results on a variety of synthetic datasets (that we did not show here for the shortage of space) indicate that the proposed algorithm yields a better fit to the circular shapes as compared to the MCS method.

VI. Conclusions

In this article we proposed a metaheuristic shell clustering technique that uses a mountain function as the objective function to be minimized with a modified Invasive Weed optimization (IWO) algorithm. Our method shows considerably good performance in extracting a variety of hyperquadric shell clusters with acceptable computation time (ranging from 15 to 30 seconds on average per run). Also it provides an efficient means to detect boundaries from edge images generated from real-life images. Our preliminary studies indicate that our method can yield more accurate prototype to fit the data-points distributed along circular shells as compared to the MCS algorithm that uses a steepest ascent algorithm. Our future research will focus on extending the concept of metaheuristic mountain clustering for the detection of other irregular shaped boundaries in two-dimensional pictures and surfaces in three-dimensional pictures.

Acknowledgement

This work was supported by the Czech Science Foundation under the grant no.102/09/1494.

References

1. J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York, Plenum, 1981.
2. R. Krishnapuram and J. Keller, “The possibilistic c-means algorithm: insights and recommendations”, *IEEE Trans. on Fuzzy Systems*, 4, 1996, 385–393.
3. R.N. Dave, “Generalized Fuzzy c-Shell clustering and detection of circular and elliptical boundaries”, Pattern Recognition, Vol. 25 (7), pp.713-721, 1992.
4. T.Balakumaran, ILA.Vennila, and C.Gowri Shankar, “Detection of microcalcification in mammograms using wavelet transform and fuzzy shell clustering”, *International Journal of Computer Science and Information Security,* Vol. 7, No. 1, 2010.
5. M.Barni, A.Mecocci and G.Perugini, “Application of possibilistic shell-clustering to the detection of craters in real-world imagery”, Proceedings of the IEEE for Geoscience and Remote Sensing Symposium, vol.1, pp.168-170, 2000.
6. R. N. Dave and K. Bhaswan, “Adaptive fuzzy C shells clustering and detection of ellipses,” *IEEE Trans. Neural Networks,* vol. 3, pp. 643-662, Sept. 1992.
7. R. Krishnapuram, H. Frigui, and O. Nasraoui, “New fuzzy shell clustering algorithms for boundary detection and pattern recognition,” in *Proc. SPIE Conf. Intell. Robots Comput. Vision X: Algorithms Techniq.,* Boston, Nov. 1991, pp. 458-465.
8. R. Krishnapuram, H. Frigui, and O. Nasraoui, “Quadratic shell clustering algorithms and their applications”, *Pattern Recog. Lett.,* vol. 14, no. 7, pp. 545-552, July 1993.
9. R. Krishnapuram, H. Frigui, and O. Nasraoui, “Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation: Parts I and 11,” *IEEE Trans. Fuzzy Syst.,* vol. 3, pp. 44-60, Feb. 1995.
10. R. R. Yager and D. P. Filev, “Approximate clustering via the mountain method”, *IEEE Trans. Systems, Man, Cybernet*. 24 (8), 1279–1284, 1994.
11. N. R. Pal and D. Chakraborty, “Mountain and subtractive clusteing method: improvements and generalization”, *Internat. J. Intell. Systems*, 15, 329–341, 2000.
12. S. L. Chiu, “Extracting fuzzy rules for pattern classification by cluster estimation. In: The 6th Internat. Fuzzy Systems Association World Congress, p. 1–4, 1995.
13. A. R. Mehrabian and C. Lucas, “A novel numerical optimization algorithm inspired from weed colonization,” *Ecological Informatics*, vol. 1, pp. 355–366, 2006.
14. A. R. Mehrabian, A. Yousefi-Koma, “Optimal positioning of piezoelectric actuators on a smart fin using bio-inspired algorithms”, *Aerospace Science and Technology*, 2007, vol. 11, pp.174–182.
15. H. Sepehri Rad, C. Lucas, “A recommender system based on invasive weed optimization algorithm”, *IEEE Congress on Evolutionary Computation*, CEC 2007, Sept. 2007, pp. 4297–4304.
16. A. R. Mallahzadeh, H. Oraizi, and Z. Davoodi-Rad, “Application of the Invasive Weed Optimization Technique For Antenna Configurations”, *Progress In Electromagnetics Research PIER 79,* 137–150, 2008.
17. A. R. Mallahzadeh, S. Es'haghi, and A. Alipour, “Design of an E-Shaped Mimo Antenna Using IWO Algorithm for Wireless Application at 5.8 GHz”, *Progress In Electromagnetics Research,* PIER 90, 187 - 203, 2009.
18. X. Zhang, Y. Wang, G. Cui, Y. Niu, and J. Xu, “Application of a novel IWO to the design of encoding sequences for DNA computing”, *Comput. Math. Appl.* 57, pp. 2001-2008, 2009.
19. A. R. Mallahzadeh, S. Es'haghi, and H. R. Hassani, Compact U-array MIMO antenna designs using IWO algorithm, *International Journal of RF and Microwave Computer-Aided Engineering*, Wiley-InterSscience, DOI: 10.1002/mmce.20379, Jul, 2009.
20. R. Gonzalez and R. Woods, *Digital image processing* (2nd ed.). Prentice-Hall Inc. 567-612, 2002.