Clustering using Particle Swarm Optimization

Nuria Gómez Blas, Octavio López Tolic

Abstract: *Data clustering has been a well-studied research field for a long time. One of the latest trends in this area is the application of Particle Swarm Optimization (PSO) in clustering which has good potential for improvements. This paper presents an approach to using Particle Swarm Optimization to cluster data. It is shown how PSO can be used to find the centroids of a user specified number of clusters. Results show that PSO clustering techniques have much potential.*

Keywords: *Social intelligence, Particle swarm optimization, Swarm computing, Clustering algorithms.*

ACM Classification Keywords*: 10010147.10010178 Computing methodologies Artificial intelligence, 10010147.10010257 Computing methodologies Machine learning, 10010147.10010257.10010293.10010294 Computing methodologies Neural networks.*

Conference topic*: Information Modelling, Information Systems, Applied Program Systems.*

MSC*: 68Q32 Computational learning theory, 68T05 Learning and adaptive systems.*

Introduction

Cluster analysis was originated in anthropology by Driver and Kroeber in 1932 and introduced to psychology by Tryon [1939] and famously used by Cattell [1943] for trait theory classification in personality psychology. Besides the term clustering, there are a number of terms with similar meanings, including automatic classification, numerical taxonomy, botryology and typological analysis. The subtle differences are often in the usage of the results: while in data mining, the resulting groups are the matter of interest, in automatic classification the resulting discriminative power is of interest. This often leads to misunderstandings between researchers coming from the fields of data mining and machine learning, since they use the same terms and often the same algorithms, but have different goals.

The notion of a cluster cannot be precisely defined, which is one of the reasons why there are so many clustering algorithms Estivill-Castro [2002]. There is a common denominator: a group of data objects. However, different researchers employ different cluster models, and for each of these cluster models again different algorithms can be given. The notion of a cluster, as found by different algorithms, varies significantly in its properties. Understanding these cluster models is key to understanding the differences between the various algorithms. Typical cluster models include:

- Connectivity models: for example, hierarchical clustering builds models based on distance connectivity.
- Centroid models: for example, the k-means algorithm represents each cluster by a single mean vector.
- Distribution models: clusters are modeled using statistical distributions, such as multivariate normal distributions used by the Expectation-maximization algorithm.
- Density models: for example, DBSCAN and OPTICS defines clusters as connected dense regions in the data space.
- Subspace models: in Biclustering (also known as Co-clustering or two-mode-clustering), clusters are modeled with both cluster members and relevant attributes.

- Group models: some algorithms do not provide a refined model for their results and just provide the grouping information.

- Graph-based models: a clique, that is, a subset of nodes in a graph such that every two nodes in the subset are connected by an edge can be considered as a prototypical form of cluster. Relaxations of the complete connectivity requirement (a fraction of the edges can be missing) are known as quasi-cliques, as in the HCS clustering algorithm.

Data clustering is a popular approach of automatically finding classes, concepts, or groups of patterns. It seeks to. partition an unstructured set of objects into clusters (groups). This implies wanting the objects to be as similar to objects in the same cluster and as dissimilar to objects from other clusters as possible. Clustering has been applied in many areas including biology, medicine, anthropology, marketing and economics. Clustering applications include plant and animal classification, disease classification, image processing, pattern recognition and document retrieval. Clustering techniques have been applied to a wide variety of research problems.

A clustering is essentially a set of such clusters, usually containing all objects in the data set. Additionally, it may specify the relationship of the clusters to each other, for example, a hierarchy of clusters embedded in each other. Clusterings can be roughly distinguished as: hard clustering – each object belongs to a cluster or not and soft clustering (also: fuzzy clustering) – each object belongs to each cluster to a certain degree (for example, a likelihood of belonging to the cluster).

Clustering algorithms can be grouped into two main classes of algorithms, namely supervised and unsupervised. With supervised clustering, the learning algorithm has an external teacher that indicates the target class to which a data vector should belong. For unsupervised clustering, a teacher does not exist, and data vectors are grouped based on distance from one another. Many unsupervised clustering algorithms have been developed. Most of these algorithms group data into clusters independent of the topology of input space. These algorithms include, among others, K-means Kanungo et al. [2002], ISODATA Memarsadeghi et al. [2007], and learning vector quantizers (LVQ) Fausett [1994].

This paper presents the fundamentals of Particle Swarm Optimization algorithm and also it introduces the application to clustering problems.

Particle swarm optimization (PSO) Kennedy and Ebehart [1995] is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

Particle Swarm Optimization Model

Particle swarm optimization (PSO) is a heuristic optimization technique, originally developed by James Kennedy and Russell C. Eberhart in 1995. PSO is considered to fall within the branch of evolutionary computing, which is commonly referred as swarm intelligence. Evolutionary Computation is a collection of methods that simulate evolution by using computers, including genetic programming, evolution strategies, Swarm Optimization, Artificial Life and other related fields. Evolutionary computation and genetic algorithms share the technique for generating populations in a stochastic way. Furthermore, this technique is used to generate new generations too.

The algorithm used by PSO is inspired by the collective work of swarms, specifically in the social organization of bird flocks and fish schools. These social organizations take on a collective behavior that comes from communication and cooperation among its members. In the algorithm, the individuals of the population are treated as particles or searching points, which have a position and velocity (the velocity vector indicates where it is headed). It move into an area of dimension n. The particle that gets better assessment is the *leader*, in the sense that the whole swarm will follow it. However any individual could change their orientation. In that case the leader passes leadership to the one that changed the orientation. The *leader* can either be global leader in the whole swarm or local in a part of it.

The emulation of these organized societies has been successful in discrete and continuous optimization problems. Initially, Kennedy and Eberhart developed a standard algorithm, which has undergone several improvements, it is a fairly simple algorithm and widely used today.

The particles move in the search space by using a combination of the best individual solution found by the particle and the best found by any of the neighboring particles. For every iteration we evaluate the performance of each particle.

The three main operations Kennedy and Ebehart [1995] algorithms use are: To assess, to compare and to imitate. Evaluation is one of the most important features of some living organisms; they evaluate learn and evolve. Besides, organisms analyze their neighbors and imitate only those who have better performance. In general, these three operations can be applied to simple social beings and to computer programs. This enable them for solvng complex problems.

Each particle i is related to three vectors: position: $x^{(i)} = (x_1, x_2 \ldots, x_d)$, the best position of its history: $p^{(i)} = (p_1, p_2 \ldots, p_d)$ and speed $v^{(i)} = (v_1, v_2 \ldots, v_d)$, given a space of dimension d. Initially values related to particles are generated randomly, then these particles move through a search space by using a system of equations that is updated for each iteration. The intention is to find the best solution. Each particle moves to the more successful neighbor, influencing the other particles. The algorithm updates the swarm at each step. It also changes the position and velocity of each particle and it applies the following rules:

$$v_d^{(i)} = v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) \tag{1}$$

$$x_d^{(i)} = x_d^{(i)} + v_d^{(i)} \tag{2}$$

$v_d^{(i)}$ is the d-th component of the velocity of particle i, $x_d^{(i)}$, the d-th component of vector position of the particle i, c_j, $(j = 1, 2)$ is a constant value (obtained from experimental results) and ϵ_1 and ϵ_2 are independent random numbers uniformly distributed in $[0.1]$. They are generated for every update, p_d is the d-th component of the improved performance of the particle in its history, and g_d is the d-th component of the particle with the best position found between neighboring particles, throughout its history. This neighborhood is defined according to the topology of the particle system. The factor c_1 is known as the factor of personal or cognitive learning and the factor c_2 as the social learning factor. Both factors have much influence on the rate of convergence of the optimization process.

Mathematical models developed for PSO vary according to how the particles interact with their neighbors, this is known as the topology of the system, which can be understood as the way the swarm of particles organizes itself. Early models of PSO used a Euclidean neighborhood. However the number of operations were high; in order to reduce the number of operations, a new mathematical model was developed. In this mathematical model neighborhoods were not related to the location of the particle; these are called local neighborhoods or *lbest models*. They can be also global or and *gbest models*. Originally *gbest* had better performance, but recent research has also shown good results in some problems when using the model *lbest* by adding some improvements to the algorithm.

In PSO, neighborhood is understood as the set of particles related to a given particle. This relationship influences the search capability and convergence. In the ring-type topology, each particle communicates only with n neighbors, $n/2$ on each side. Ring topology presented is the simplest: $n = 2$. This means that only two neighboring particles are evaluated. In the wheel topology all information is concentrated in a central particle. In the star type each particle is related to all the particles of the swarm.

The standard Particle Swarm Optimization schema is described as follows, see algorithm 1.

It begins with a population of particles with position and velocity randomly assigned in the search space. With regard to calculating the velocity of the particles, we consider a a maximum value as restriction. This is called V_{max}. This is done because an explosion might happen, since the velocities could be increased quickly. The selection of the

Algorithm 1 Standard Particle Swarm Optimization Algoritm

```
1: Create initial population of particles $x^i$
2: loop
3:   for each particle of the swarm do
4:     if $f(x) > f(p)$ then
5:       for $d = 1 \to D$ do
6:         $p_d = x_d$
7:       end for
8:     end if
9:     $g = i$
10:    for $j \in J$ do
11:      if $f(p_j) > f(p_g)$ then
12:        $g = j$
13:      end if
14:    end for
15:    for $d = 1 \to D$ do
16:      $v_d(t) = v_d(t-1) + c_1\epsilon_1(p_d - x_d(t-1)) + c_2\epsilon_2(g_d - x_d(t-1))$ [1]
17:      $x_d = x_d + v_d$
18:    end for
19:  end for
20: end loop
```

values V_{max} is difficult to determine. They will be chosen by trial and error. In very large spaces, large values are usually selected in order to ensure proper exploration. This is justified since a large inertia weight facilitates exploration in new areas in the global search space, while a small one facilitate the exploration in a local area. In the case of small spaces, small values are required to prevent the explosion. As for the size of the particle population, empirical results have shown that good results are not always obtained by increasing the number of particles of the population. Some examples have been influenced by that but others have not.

The particle swarm is actually more than just a collection of particles. A particle by itself has almost does not solve any problem; progress takes place only when they i.e. the particles interact. Populations are organized according to some sort of communication structure or topology. This is often thought of as a social network. The topology typically consists of bidirectional edges connecting pairs of particles. It is like the alphabet j appearing in i's neighborhood, and likewise i in j's neighbour. Each particle communicates with other particles and is affected by the best point found by any member of its topological neighborhood.

Model variants

Regarding the PSO algorithm, different variants have been developed, aimed at speeding up the convergence of it. In addition to the unconstrained optimization problem in discrete or continuous variable, the multi target problem and the constrained problem have been addressed. We have also developed hybrid optimization techniques, PSO technique has been tested with good results for training Artificial Neural Networks. When applying the method of Back Propagation, we are able to find appropriate weights that minimize an error function through a succession of iterations. On the other hand, by applying the PSO technique, the weights found are more efficient just by making small modifications to the algorithm. The new guidelines are aimed at avoiding PSO stagnation of the local optimal solutions.

Kennedy et al. [2001] proposed adjustments to the velocities of the particles by using a factor w called *inertial weight*. This factor utilizes the inertia of the particles in the process of friction when they are moving. This modification in the algorithm is done to control the search space. In order to do that it must change (3). The large inertia weight makes the global search easier; however small inertia weight does not improve local search. That is why was the initial value is greater than 1.0 to promote global exploration, and then gradually decreases to obtain more refined solutions. The algorithm decreases linearly at each iteration. Moreover, the use of inertial weight removes the restriction V_{max} on the velocity.

$$v_d^{(i)} = wv_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) \tag{3}$$

In each iteration, inertia weight decrease linearly through the following expression:

$$w = w_{max} - (w_{max} - w_{min})\frac{g}{G} \tag{4}$$

g is the index of the generation, G is the maximum number of iterations previously determined, w_{max} is a value greater than 1, and w_{min} a value under 0.5. This variation of the method has proven to accelerate convergence.

Clerc and Kennedy [2002] obtain another variation in the speed calculation. A constriction factor χ is introduced with that purpose, This factor depends on the constants that are used when calculating speed. This factor affects the formula (1) The aim is to avoid the explosion of velocity:

$$v_d^{(i)} = \chi[v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)})] \tag{5}$$

χ is:

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \quad \varphi = c_1 + c_2 = 4.1$$

The results are: $\chi = 0.729$ and c_1=c_2= 2.05. These parameters were obtained from performing several tests.

χ factor is similar to the inertial weight. This means that controlling the velocity with $V_(max)$ is not required when χ is used. Bratton and Kennedy [2007], analyzed the stability of this algorithm by using these values and by following a comparative study of both PSO algorithms (inertial weight and χ factor). Both of them are mathematically equivalent, in particular the algorithm with constriction factor is a special case of the inertial weight. Moreover, Parsopoulos et al. [2001], combined both for problems with constraints and they obtained equally good results in several tests.

We observe that the convergence always becomes slower when problem size increase, so when it comes to high-dimensional problems, a larger number of iterations occurs. Researchers Uchitane and Hatanaka [2015]; Hatanaka et al. [2015] developed a PSO model, where velocity values are updated, by considering the rotation of the coordinate system. This model is aimed at problems of high dimensionality and it showed good results when applied to all functions of De Jong, (larger dimensions).

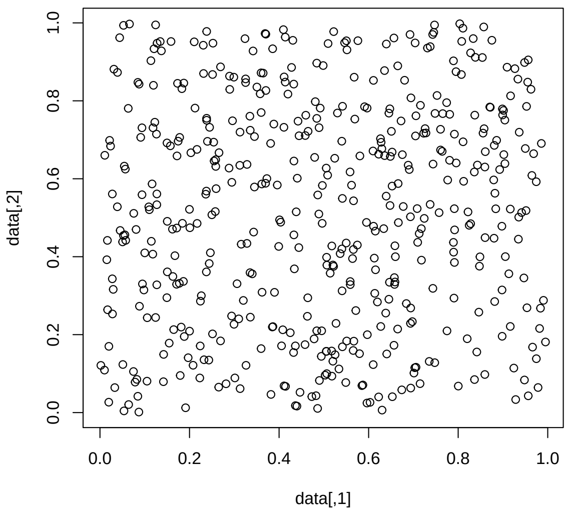
Clustering using Particle Swarm Optimization

This section shows a first approach to data clustering problems using Particle Swarm Optimization. Pictures have been generated using R, `Sweave` and LaTeX. Algorithms implemented in this paper can be shown in listings 1 and 2.

Data sets have been generated using an uniform distribution in square [0,1] with `runif` R function. Figure 1 shows used data in the clustering problem represented by figure 2.

In order to start with the PSO algorithm each particle is considered to represent the centroid of a given cluster. In the context of clustering, a single particle represents the cluster centroid vectors, $p = (x, y)$ in the 2 dimensional space. When dealing with multiple clusters, let's say n, each particle is represented as a collection of clusters, that is $p = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$. Using this approach particle dimension will increase when working according the number of clusters.

Figure 1: Uniform data in $[0, 1]$ used in the PSO clustering problem.



The fitness of panicles is easily measured as the quantization error. The fitness function of the data clustering problem is given as follows:

$$f = \frac{1}{1+d} \tag{6}$$

The function f should be minimized. Where

$$d \text{ is the mean distance of data with respect to clusters} \tag{7}$$

One particle in the swarm represents one possible solution for clustering. Therefore, a swarm represents a number of candidate clustering solutions for the data set. At the initial stage, each particle randomly chooses k different data set from the collection as the initial cluster centroid vectors.

Figure 2: Clustering uniform data in $[0, 1]$ square using PSO with different clusters.

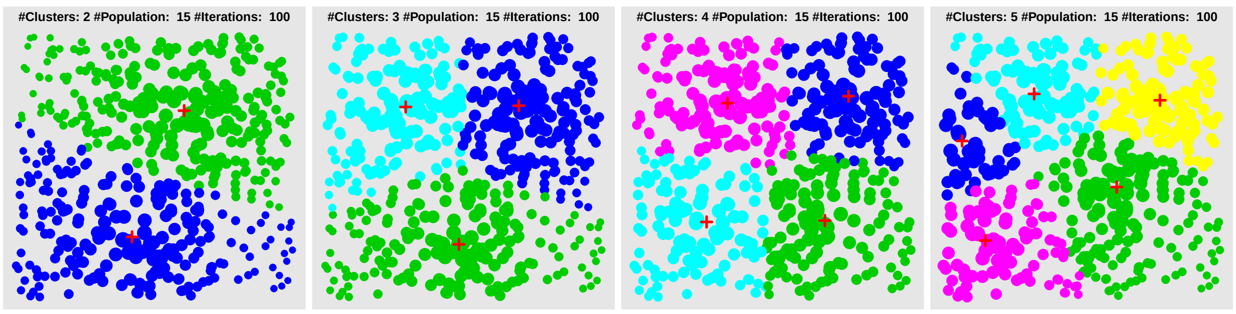


Figure 2 shows obtained results with the PSO algorithm using different clusters, that is 2, 3, 4, and 5 clusters. We can observe that all centroid are well distributed.

Figures 3 and 4 shows other different distribution of initial data using different clusters.

Final Remarks

One shortcoming of the PSO algorithm is the formation of number of small clusters which was overcome by introducing a one setp K-means operator that forces the small clusters into the bigger ones and finishes the

Figure 3: Clustering uniform data using PSO with different clusters.

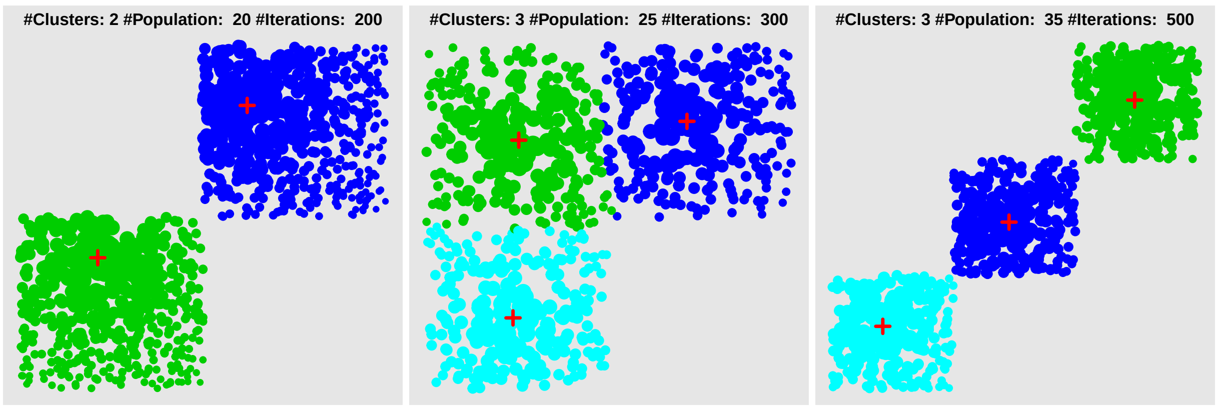
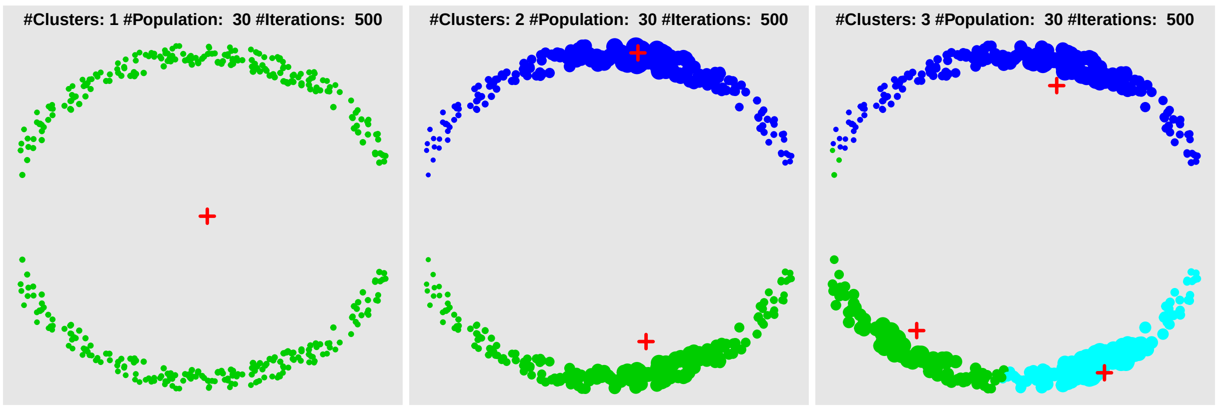


Figure 4: Clustering circle data using PSO with different clusters.



clustering operation van der Merwe and Engelbrecht [2003]. But this goes against the spirit of the algorithm and the user will have to specify the minimum number below which the cluster will be considered as a small cluster. Another approach could be to introduce a second phase of PSO in which the parameters are reinitialized and positions reallocated. New force functions could be defined which will increase the attractive force between larger and smaller clusters and reduces the force between larger clusters. Particle swarm optimization provides number of future applications and improvements in the domain of clustering.

Appendix: Listings of PSO clustering in R

Listings 1 and 2 show the implementation algorithms used in this paper. First one –listing 1– correspond to a general PSO algorithm while the second one –listing 2– corresponds to the clustering approach based on a PSO algorithm.

Listing 1: Basic Particle Swarm Algorithm. File `pso.R`.

```
fitness <- function(pop, calculate_fitness) {
  pop[,dimensions+1] <- calculate_fitness(pop[,1:dimensions]);
  return(pop)
}

pso <- function(verbose, individuals, dimensions, iterations, c1, c2, inertia, calculate_fitness) {
  if (verbose)
    cat ("Startint PSO\n");

  #########################
  # Init population (actual + fitness + local_best + fitness)
  population <- array (runif(2*(dimensions+1)*individuals), c(individuals, 2*(dimensions+1)));

  # Calculate fitness
  population <- fitness (population, calculate_fitness);
  # Init local_best
  population[,(dimensions+2):(2*(dimensions+1))] <- population[,1:(dimensions+1)];
  # Init global_best
  global_best <- population[max.col(t(population[,dimensions+1])),1:(dimensions+1)];

  # Init velocity
  velocity <- array (rnorm(individuals*dimensions), c(individuals, dimensions));
  inertia_coeff <- inertia/iterations;

  # Init historic data
  fitness_historic <- array(0, iterations);
  global_best_historic <- array(0, c(iterations,dimensions + 1));

  for (iteration in 1:iterations) {
    uniform_coeffs <- runif(individuals*2);
    # Update velocity
    velocity <- inertia * velocity +
      c1 * uniform_coeffs[1:individuals] * (population[,(dimensions+2):(2*(dimensions+1)-1)] - population[,1:dimensions
           ]) +
      c2 * uniform_coeffs[(individuals+1):(2*individuals)] * (global_best[1:dimensions] - population[,1:dimensions]);
    # Update inertia
    inertia <- inertia - inertia_coeff;

    # Update population with fitness
    population[,1:dimensions] <- population[,1:dimensions] + velocity;
    population <- fitness (population, calculate_fitness);

    # Update local_best
    for (individual in 1:individuals) {
      if (population[individual, dimensions+1] > population[individual, 2*(dimensions+1)]) {
        population[individual,(dimensions+2):(2*(dimensions+1))] <- population[individual,1:(dimensions+1)];
      }
    }

    # Update global_best
    if (max (population[,dimensions+1] > global_best[dimensions+1])) {
      global_best <- population[max.col(t(population[,dimensions+1])),1:(dimensions+1)];
    }

    if (verbose)
      cat("Iteration ", iteration, " Fitness ", global_best[dimensions+1], "\n");

    # Keep historic fitness data
    fitness_historic[iteration] <- global_best[dimensions+1];

    # Keep historic global_best
    global_best_historic[iteration,] <- global_best;
  }

  ########################
  if (verbose)
    cat ("Finish PSO\n");

  return(list(best=global_best, fitness_evol=fitness_historic, global_best=global_best_historic));
}
```

Please note that in listing 2 parameters `iterations`, `individuals`, `clusters` and `data` must be set to the desired configuration to solve a given problem. In our case some sample code to call these functions is the following, `Sweave` package and R have been used to generate a source LaTeX:

```
data <- array(runif(1000), c(500,2),dimnames=list(NULL,paste('Dimension',1:2)));
individuals <- 15;
iterations <- 100;
clusters <- 2;
source('nclustering.R');
```

Listing 2: Clustering Algorithm using PSO. File `nclustering.R`

```
##### N-Clustering sample
source('pso.R')

##### global parameters
# iterations <- 1;
# individuals <- 20;
# clusters <- 1;
# data <- array(runif(1000)*2 -1 , c(500,2));


##### general parameters
clusters_dimension <- 2;
dimensions <- clusters_dimension*clusters;
c1 <- 2;
c2 <- 1;
inertia <- .8;

# fitness function in clustering process
clustering_fitness <- function(ind) {
  val <- array(0, c(length(ind[,1]),1));
  for(individual in 1:length(ind[,1])) {
    dist <- 0;
    for(i in 1:length(data[,1])) {
      dist_vector <- (data[i,]-ind[individual,])^2;
      dist_min <- array(0, clusters);
      for (j in 1:clusters) {
        dist_min[j] <-  sqrt(sum(dist_vector[((j-1)*clusters_dimension + 1):(j*clusters_dimension)]));
      }
      dist <- dist + min(dist_min)/length(data[,1]);
    }
    val[individual] <- 1 / (1 + dist);
  }
  return(val)
}

res <- pso(FALSE, individuals, dimensions, iterations, c1, c2, inertia, clustering_fitness);

# setup margins and title size
par(mar=c(0.2,0.2,2,0.2),bg=rgb(0.9,0.9,0.9), cex.main=1.5);

# plot data
plot(data[,1], data[,2], cex= 0.1, xlab='', ylab='',lwd =1, axes=FALSE);
title(paste('#Clusters:', clusters, '#Population: ', individuals, '#Iterations: ', iterations));

distribution <- array(0,c(1,clusters), dimnames = list('Number of elements:',paste('Cluster',1:clusters)));

for(i in 1:length(data[,1])) {
  dist_vector <- (data[i,] - res$best[1:dimensions])^2;
  dist_min <- array(0, clusters);
  for (j in 1:clusters) {
    dist_min[j] <-  sqrt(sum(dist_vector[((j-1)*clusters_dimension + 1):(j*clusters_dimension)]));
  }
  dist_min <- 1 / (1 + dist_min);
  winner <-  max.col(t(dist_min));
  distribution[winner] <- distribution[winner] + 1;
  points(data[i,1], data[i,2],lwd=2,cex=(4*max(dist_min)^2), pch=19, col = winner+2);
}

for (i in 1:clusters)
  points(res$best[((i-1)*clusters_dimension)+1], res$best[(i)*clusters_dimension], col='red', pch = 3,lwd=5, cex=2);
```

Bibliography

Bratton, D. and Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium, SIS 2007, Honolulu, Hawaii, USA, April 1-5, 2007*, pages 120–127.

Cattell, R. (1943). *The Description of Personality: Basic Traits Resolved Into Clusters*. American psychological association.

Clerc, M. and Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evolutionary Computation*, 6(1):58–73.

Estivill-Castro, V. (2002). Why so many clustering algorithms: A position paper. *SIGKDD Explor. Newsl.*, 4(1):65–75.

Fausett, L., editor (1994). *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Hatanaka, T., Chopra, N., and Fujita, M. (2015). Passivity-based bilateral human-swarm-interactions for cooperative robotic networks and human passivity analysis. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*, pages 1033–1039.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., Wu, A. Y., Member, S., and Member, S. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:881–892.

Kennedy, J. and Ebehart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948.

Kennedy, J., Eberhart, R., and Shi, Y. (2001). *Swarm Intelligence*. Morgan Kauff-man.

Memarsadeghi, N., Mount, D. M., Netanyahu, N. S., and Moigne, J. L. (2007). A fast implementation of the isodata clustering algorithm. *Int. J. Comput. Geometry Appl.*, 17(1):71–103.

Parsopoulos, K., Plagianakos, V. P., and Magoulas, G. D. (2001). Stretching technique for obtaining global minimizers through particle swarm optimization. In *Proceedings of Particle Swarm Optimization Workshop*, pages 22–29.

Tryon, R. (1939). *Cluster Analysis: Correlation Profile and Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality*. Edwards brother, Incorporated, lithoprinters and publishers.

Uchitane, T. and Hatanaka, T. (2015). A study on multi-objective particle swarm model by personal archives with regular graph. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 2685–2690.

van der Merwe, D. W. and Engelbrecht, A. P. (2003). Data clustering using particle swarm optimization. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 1, pages 215–220 Vol.1.

Authors' Information

Nuria Gómez Blas *- Dept. de Sistemas InformÃątcos, Escuela Técnica Superior de Sistemas Informáticos, Universidad Politécnica de Madrid, Crta. de Valencia km. 7, 28031 Madrid, Spain; e-mail: nuria.gomez.blas@upm.es*
Major Fields of Scientific Research: Bio-inspired Algorithms, Natural Computing

Octavio López Tolic *- Dept. de Sistemas InformÃątcos, Escuela Técnica Superior de Sistemas Informáticos, Universidad Politécnica de Madrid, Crta. de Valencia km. 7, 28031 Madrid, Spain; e-mail: oa@etsisi.upm.es*
Major Fields of Scientific Research: Bio-inspired Algorithms, Natural Computing