A GRASP algorithm for Clustering

J.R. Cano[1], O. Cordón[2], F. Herrera[2], and L. Sánchez[3]

[1]Dept. of Software Engineering
University of Huelva, 21071 La Rabida (Huelva), Spain
Jose.cano@diesia.uhu.es
[2] Dept. of Computer Science and A.I.
University of Granada, 18071-Granada, Spain
{ocordon,herrera}@decsai.ugr.es
[3] Dept. of Computer Science
University of Oviedo, Oviedo, Spain

Abstract. We present a new approach for Cluster Analysis based on a Greedy Randomized Adaptive Search Procedure (GRASP), with the objective of overcoming the convergence to a local solution. It uses a probabilistic greedy Kaufman initialization for getting initial solutions and K-Means algorithm as a local search algorithm. We compare it with some typical initialization methods: Random, Forgy, Macqueen and Kaufman. The new approach obtains high quality solutions for the benchmark problems.

1. Introduction

Clustering is a basic process to human understanding. The grouping of related objects can be found such diverse fields as statistics, economics, physics, psychology, biology, pattern recognition, engineering, and marketing [3,5].

The Clustering problem involves partitioning a set of entities into a given number of subsets and finding the location of a centre for each subset in such a way that a dissimilarity measure between entities and centres is minimized. K-Means is one of the most known clustering algorithms. Although it is known for its robustness, it can fall in local optimal solutions easily. It is widely reported that the K-Means algorithm suffers from initial starting conditions effects: initial clustering and instance order as shown in [4].

In this contribution, we propose a Greedy Randomized Adaptive Search Procedure (GRASP) [2] applied to the Clustering problem, using K-Means as local search procedure. Our algorithm tries to eliminate the classical problem of the K-Means algorithm, hold up solutions in local optima, by permitting a higher exploration and exploitation of the search space, with a medium computational cost.

In order to do that, the paper is organized as follows. Section 2 introduces a background on clustering, K-means and initialization approaches. Section 3 presents the GRASP approach to clustering. Section 4 shows the experiments and their analysis, and finally, Section 5 points out some concluding remarks.

2. Background

A common problem in cluster analysis is partitioning objects into a fixed number of groups to optimize an objective function-based clustering. These objects are measured along several features that characterize them. Patterns can be viewed as vectors in a high dimensional space, where each dimension corresponds to one feature.

In this section we introduce the formalization of clustering, the K-Means algorithms, and four initialization approaches.

2.1. Clustering Problem

The clustering problem can be formalized as follows [1]: Considering N entities e_i, each with an associated weight w_i (i=1,..,N), we search for k centres c_j (j=1,...,k) minimizing:

$$f(c_1,...,c_k) = \sum_{i=1}^{N} \min_j \left(w_i d\left(e_i, c_j\right)\right)$$

where $d(e_i,c_j)$ measures the dissimilarity between e_i and c_j. In our case, where the entities are described by their co-ordinates in $\Re^m$, $d(e_i,c_j)$ is the Euclidean distance.
Basically, clustering is a combinatorial optimization problem.
Let

Q be set containing all objects to be clustered,
C be the set of all feasible clustering of Q,
J: C $\rightarrow \Re$ be the internal clustering criterion,

then the problem involves

Minimize J(c) subject to c $\hat{I}$ *C.*

The complexity of the clustering problem is given by different factors:

1. The clustering is an NP-HARD problem. Therefore, an exhaustive approach is not practicable due to the exponential number of the potential partitions of the input data. The number of possible partitions of N elements into k clusters is given by

$$\prod(k,N) = \frac{1}{k!}\sum_{j=1}^{k}(-1)^{k-j}\begin{pmatrix} k \\ j \end{pmatrix}(j)^N .$$

2. The clustering complexity grows if the number of groups is unknown. In such a case he number of solutions becomes:

$$\prod(k,N)=\sum_{i=1}^{k}\frac{1}{i!}\sum_{j=1}^{i}(-1)^{i-j}\binom{i}{j}(j)^{N}.$$

3. It is very difficult to translate the concept of 'similarity' into a unique mathematical model, but this depends on the clustering goal.

Due to these reasons, trying to get a global optimal solution by means of an efficient and robust method is difficult. Thus, there is a considerable interest in the design of new heuristics to solve large-sized practical clustering problems.

2.2. K-Means Algorithm

K-means evaluates a set of k selected objects, which are considered representatives for the k clusters to be found within the source set of N objects. Given the set of representative objects, the remaining objects are assigned to the nearest representative one, using a chosen distance measure.

The philosophy is that a better set of clusters is obtained when the k representative objects are more centrally located in the cluster they define. For this reason, a suitable objective function to be minimized is the sum of the distances from the respective centers to all the other objects of the same cluster. The function value depends on the current partition of the database $\{C_1,...,C_k\}$:

$$J:\prod_k(\Omega)\rightarrow\Re$$

with $\pi_k(\Omega)$ being the set of all partitions of the database $\Omega=\{e_1,...,e_N\}$ in k non-empty clusters. Each instance e_i of the N instances in the database Ω is an m-dimensional vector.

The K-Means algorithm finds locally optimal solutions using the Euclidean distance in the clustering criterion. This criterion is sometimes referred to as square-error criterion. Therefore, it follows that:

$$J(\{C_1,...,C_k\})=\sum_{i=1}^{k}\sum_{j=1}^{k_i}\left\|e_{ij}-c_i\right\|$$

where k is the number of clusters, k_i the number of objects of the cluster i, e_{ij} is the j-th object of the i-th cluster and c_i is the centroid of the i-th cluster defined as:

$$c_i=\frac{1}{k_i}\sum_{j=1}^{k_i}e_{ij},\ i=1,...,k$$

The pseudo-code for this algorithm is:

```
1. Select an initial partition of the database in $k$ clusters $\{C_1,...,C_k\}$
2. Calculate cluster centroids, using the expression of its definition.
```

3. For every e_i in the database and following the instance order DO
   - Reassign instance e_i to its closest cluster centroid. Hence, $e_i \in C_s$ is moved to C_t if $||e_i - c_t|| \leq ||e_i - c_s||$ for all t=1,...,k, t≠s.
   - Recalculate centroids for those clusters.
4. If cluster membership is stabilized then stop else go to *step 3*.

The K-Means algorithm has the drawbacks:

- It assumes that the number of clusters k in the database is known, which is not necessarily true.
- It is especially sensitive to initial starting conditions: initial clusters and instance order.
- It converges finitely to a local minima, defined by a deterministic mapping from the initial conditions to the final solution.

2.3. Initialization Approaches

The second problem, sensitivity to initial conditions, may be mitigated using different values of *k*, some instance orders and different initialization methods.

In this section we describe four initialization approaches [4]. Each one generates *k* initial clusters following some kind of heuristic and produces a different K-Means response.

These four classical methods are:

– *Random*: Divides the database into a partition of K randomly selected clusters.
– *Forgy: k* instances of the database (seeds) are chosen at random and the rest of the instances are assigned to the cluster represented by the nearest seed.
– *Macqueen: k* instances of the database (seeds) are chosen at random. Following the instance order, the rest of the instances are assigned to the cluster with the nearest centroid. After each assignment, a recalculation of the centroid has to be carried out.
– *Kaufman:* The initial clustering is obtained by the sucessive selection of representative instances. The first representative is chosen to be the most centrally located instance in the database. The rest of representative instances are selected according to the heuristic rule of choosing the instances that promise to have around them a higher number of the rest of instances and that are located as far as possible from the previously determinated ones.

Differences between these initialization methods are:

- Random and Forgy generate an initial partition independently of the instance order.
- Macqueen generates an initial partition that depends on the instance order.
- Kaufman is the only deterministic one, based on a greedy approach.

3. GRASP Approach to the Clustering Problem

In this section we introduce the GRASP approach and present its application to clustering.

3. 1. Greedy Randomized Adaptive Search Procedure (GRASP)

A generic GRASP pseudo-code is shown as follows [2]:

```
Procedure grasp( )
  InputInstance();
  For GRASP stopping criterion not satisfied
     ConstructGreedyRandomizedSolution(Solution);
     LocalSearch(Solution);
     UpdateSolution(Solution,BestSolutionFound);
  Rof
  Return(BestSolutionFound);
End grasp;
```

In the construction phase, a feasible solution is iteratively constructed, choosing one element at a time. At each construction algorithm iteration, the choice of the next element to be added is determined by ordering all elements in a candidate list with respect to a greedy selection function. This function measures the benefit of selecting each element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top one. The list of the best candidates is called the restricted candidate list (RCL) and has dimension *l*. This choice technique allows different solutions to be obtained at each GRASP iteration, but does not necessarily compromise the power of the adaptive greedy component of the method.

The GRASP construction phase pseudo-code is:

```
Procedure ConstructGreedyRandomizedSolution(Solution)
   Solution={};
   For Solution construction not done
      MakeRCL(RCL);
      s = SelectElementAtRandom(RCL);
      Solution = Solution ∪ {s};
      AdaptGreedyFunction(s);
   Rof
End ConstructGreedyRandomizedSolution;
```

The solutions generated by a GRASP construction algorithm are not guaranteed to be locally optimal with respect to simple neighborhood solutions. Hence, it is almost always benefitial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by sucessively replacing the current solution by a better solution in the neighborhood of the current one. The key to success for a local search algorithm involves a suitable choice for a neighborhood structure, efficient neighborhood search techniques, and the starting solution.

Finally, the GRASP local search phase pseudo-code is shown:

```
Procedure Local (P,N(P),s)
   For s not locally optimal
       Find a better solution $t \in N(s)$;
       Let s=t;
   Rof
   Return (s as local optimal for P)
End local
```

with *P* being a problem (clustering in our case), *N(P)* being a mechanism to obtain neighbours for *P*, and *s* being a solution for *P*.

3.2. A proposal on GRASP for the Clustering Problem

Following the generic GRASP structure, it is easy to adapt the algorithm to the clustering problem using the greedy Kaufman method.

The stopping criterion is defined in terms of the maximum number of iterations, whilst the K-Means is introduced as a local search algorithm.

To construct the greedy randomized solution, the Kaufman initialization [3] is taken as a base, because it is a greedy deterministic initialitation algorithm. Using the Kaufman criterion, the RCL list is generated by the most promising objects for each center of the solution and one of those candidates is randomly selected.

A generic pseudo-code of the GRASP construction phase for clustering is:

```
Step 1. Select the most centrally located instance as the first seed.
Step 2. FOR every non selected instance $e_i$ DO
      Step 2.1.FOR every non selected instance $e_j$ DO
         Calc $C_{ji} = \max(D_j - d_{ji}, 0)$ where $d_{ji} = ||e_i - e_j||$
         and $D_j = \min_s d_{sj}$ being $s$ one of the
         selected seeds
      Step 2.2. Calculate the gain of selecting $e_i$
            by $\Sigma_j C_{ji}$
Step 3. MakeRCL(RCL) by selecting the $l$ instances $e_i$ which maximizes $\Sigma_j C_{ji}$
Step 4. SelectElementAtRandom(RCL)
Step 5. If there are k selected seeds THEN stop ELSE go to Step 2.
Step 6. For having a clustering assign each nonselected instance to the cluster
        represented by the nearest seed.
```

When the *k* objects have been taken, the local search (K-Means in this case) is applied taking the clustering obtained as initialization. Then, we compare the local solution cost obtained with the best solution cost found so far, and we take the best. This process continues until all the iterations have been done.

The use of K-Means offers an efficient and low computational cost method to obtain relatively good solutions, but converges to a local minimum. The GRASP construction phase corrects this problem, performing a wide space exploration search. Note that our algorithm constitutes a new initialization method for K-Means, which better explores the search space.

4. Experimental Results

Next, we present the sample process, the results obtained in our experiments, and an analysis.

4.1. Sampling Process

The performance of our algorithm is studied with various instance sets, trying to get conclusions independent of the problem. Four real-world databases are considered:

- Glass, which has 214 instances, 9 attributes and 7 clusters that can be grouped in 2 bigger classes.
- Titanic, which has 2200 instances, 4 attributes and 2 classes.
- Segmentation Image, which has 2310 instances, 19 attributes and 7 classes.
- Pima, which has 768 instances, 8 attributes and 2 classes.

Since the K-Means algorithm strongly depends on initial conditions, this problem of initial starting conditions is mitigated using different values of *k* and some instance orders.

The following initial number of clusters have been considered:

- Glass: k= 7, 10
- Titanic: k= 4, 10
- Segmentation Image: k= 6, 12
- Pima: k= 6, 10

First, we applied three K-Means variants (each one with its own initialization: Random, Forgy and Macqueen). The sampling process followed is based on the combination of four initial partitions and four instance orders (see [4]) and taking the best result of those sixteen runs.

This process is executed ten times and the following values are taken: arithmetic mean, standard deviation and the best solution.

On the other hand, the evaluation of K-Means with Kaufman initialization has been done using 10 random instance orders of each problem. Although Kaufman initialization seems to be a completely deterministic initialization method, this is not completely true. When one instance has the same Euclidean distance to two cluster centers, this instance will be associated to the first of them. If we have some instances in this situation, their order will modify the evolution of the *k* centers.

The proposed GRASP algorithm is studied using sixteen iterations in each execution (running sixteen K-means). Ten executions of the GRASP algorithm are made, getting the arithmetic mean, the standard deviation and the best solution from those executions. The RCL size used is fifteen (*l* = 15), which is flexible enough to obtained optimal solutions due the database sizes. Therefore, in order to compare GRASP evaluation with the remaining K-Means initialization methods, we use the same number of K-Means runs than GRASP iterations for every execution of Random, Forgy and Macqueen initialization methods.

4.2. Results

Experimental results are presented in Tables 1-8.

Table 1. Glass results, k=7

	Arithmetic mean	Standard Deviation	Best Solution
Random	206.632	0.704	205.307
Forgy	206.493	0.723	205.889
Macq.	206.635	0.676	205.889
Kaufm.	211.658	0	211.158
GRASP	205.239	0.486	204.992

Table 2. Glass results, k=10

	Arithmetic mean	Standard Deviation	Best Solution
Random	179.042	2.670	175.945
Forgy	176.710	1.205	176.044
Macq.	177.144	1.389	176.044
Kaufm.	188.691	0	188.691
GRASP	176.058	0.089	175.945

Table 3. Titanic results, k=4

	Arithmetic mean	Standard Deviation	Best Solution
Random	1110.034	19.662	1080.426
Forgy	1437.635	297.603	1282.327
Macq.	1442.327	301.718	1282.327
Kaufm.	1170.012	0	1170.012
GRASP	1071.458	1.368	1070.661

Table 4. Titanic results, k=10

	Arithmetic mean	Standard Deviation	Best Solution
Random	880.530	71.877	670.166
Forgy	992.681	82.042	930.914
Macq.	991.914	84.564	930.914
Kaufm.	340.696	0	340.696
GRASP	329.234	4.196	327.234

Table 5. Segmentation Image results, k=6

	Arithmetic mean	Standard Deviation	Best Solution
Random	158581.932	470.346	158141.047
Forgy	158261.497	626.579	157923.297
Macq.	158334.78	321.169	157923.297
Kaufm.	159233.335	0.0178	159233.297
GRASP	158246.504	312.505	157917.547

Table 6. Segmentation Image results, k=12

	Arithmetic mean	Standard Deviation	Best Solution
Random	117819.845	3263.117	114229.797
Forgy	114244.441	7.570	114237.953
Macq.	115645.357	8.724	114237.842
Kaufm.	114248.172	0	114248.172
GRASP	114221.817	1.066	114220.75

Table 7. Pima results, k=6

	Arithmetic mean	Standard Deviation	Best Solution
Random	29415.066	15.681	29403.762
Forgy	29426.369	14.519	29403.762
Macq.	29428.447	15.117	29403.762
Kaufm.	30729.2	0.013	30729.195
GRASP	29403.763	0.001	29403.762

Table 8. Pima results, k=10

	Arithmetic mean	Standard Deviation	Best Solution
Random	24287.956	134.991	24120.801
Forgy	24136.125	26.718	24117.748
Macq.	24137.881	23.687	24117.748
Kaufm.	24340.811	0	24340.811
GRASP	24127.770	7.929	24117.748

4.3. Analysis

Comparing results, we notice that Forgy and Macqueen usually give very similar results. They seem to have the same behaviour, keeping the same or similar local minima. Their results improve those given by Random initialization, with any number of clusters, except in the Titanic database. Thus, both Forgy and Macqueen initialization respond better than Random, with this difference more significant when *k* grows.

It is clear that Kaufman initialization by itself does not obtain the best results compared to the classical initialization methods. We conclude that the heuristic used in Kaufman initialization needs a more flexible centroid selection to reach this global optimum.

Finally, we compare the results obtained from GRASP with the remaining K-Means initialization methods. As said, this comparison is feasible because we are comparing ten GRASP runs versus ten sampling processes runs (16 K-Means runs with different initialization approaches). This comparison is based on effectiveness and robustness:

- If we study each table we notice that GRASP gets the best results in every problem for all initial numbers of clusters k. GRASP is the most effective algorithm, because it is doing a better exploration of the search space than K-Means using Kaufman initialization can do by itself.
- On the other hand, GRASP presents similar robustness (small values for the standard deviation) to Kaufman K-Means in most of the cases, and better robustness than the remaining approaches.

Although GRASP is computationally a little more demanding than the remaining K-Means initializations with the same number of runs/iterations (due to the computational time needed by the construction phase), GRASP induces the K-Means algorithm to present a better performance and a more robust behaviour.

5. Concluding Remarks

The K-Means algorithm suffers from its dependence on initial conditions which under certain conditions leads to a local minimum. We present a GRASP algorithm based on Kaufman initialization to avoid this drawback.

The GRASP algorithm has been empirically compared with four classical initialization methods (Random, Forgy, Macqueen and Kaufman) being the most effective algorithm.

References

1. M.R. Anderberg, Cluster Analysis and Applications, Academic Press, 1973.
2. T.A. Feo and M.G.C. Resende, Greedy Randomized Adaptive Search Procedure, Journal of Global Optimization 2 (1995) 1-27.
3. L. Kaufman and P.J. Rousseeuw, Finding Groups in Data. An Introduction to Cluster Analysis. Wiley, 1990.
4. J.M. Peña, J.A. Lozano, and P. Larrañaga, An empirical comparison of four initialization methods for the K-Means algorithm. Pattern Recognition Letters 20 (1999) 1027-1040.
5. S. Theodoridis and K. Koutroumbas, Pattern Recognition. Academic Press, 1999.