Dynamic local search for clustering with unknown number of clusters

Ismo Kärkkäinen and Pasi Fränti
Department of Computer Science, University of Joensuu
Box 111, FIN-80101 Joensuu, FINLAND

Abstract

Dynamic clustering problems can be solved by finding several clustering solutions with different number of clusters, and by choosing the one that minimizes a given evaluation function. This kind of brute force approach is general but not very efficient. We propose a new dynamic local search that solves the number and location of the clusters jointly. The algorithm uses a set of basic operations, such as cluster addition, removal and swapping. The clustering is found by the combination of trial-and-error approach of local search, and the local optimization capability of the GLA. The algorithm finds the results 30 times faster than the brute force approach.

Keywords: *clustering, number of clusters, vector quantization, optimization.*

1. Introduction

Clustering is an important problem that must often be solved as a part of more complicated tasks in pattern recognition, image analysis and other fields of science and engineering [1, 2]. It aims at answering two main questions: *how many clusters* there are in the data set and *where* they are located. We denote the problem here as *static clustering* if the number of clusters is known beforehand, and as *dynamic clustering* if the number of clusters must also be solved.

Static clustering problem can be solved by methods such as the *Generalized Lloyd algorithm* (GLA) [3], *simulated annealing* [4], *deterministic annealing* [5], *genetic algorithm* [6] among many others. *Randomized Local Search* (*RLS*) is a good choice for the clustering because of its competitive performance according to the results [7] in terms of optimizing the evaluation function value. Its simplicity makes it easy to generalize for the dynamic clustering problem.

The RLS method can also be generalized to the case where the number of clusters is unknown. The method is applied to every reasonable number of clusters and the correct solution is the one that minimizes the given optimization function. This method (referred to as *Brute Force*) is general but inefficient. A more efficient approach (referred to as *stepwise local search*) is to utilize the previous clustering (with m clusters) when solving the current one (with m+1 clusters), and by defining appropriate stopping criterion for the iterations [8]. The algorithm still uses most of the time for optimizing solutions with completely wrong number of clusters.

In this paper, we propose a more efficient approach called *dynamic local search* (DLS). It optimizes the number and the location of the clusters jointly. The main motivation is that most of the computation should be spent on solutions with the correct, or nearly correct number of clusters. We first derive a set of basic operations *cluster addition, cluster removal* and *cluster swapping*. We then study how the operations should be applied in order to achieve the correct clustering in most efficient way.

The main problem in the dynamic local search is the following. In static clustering, the RLS can find the correct clustering starting from any initial solution. The longer the algorithm is iterated, the likely if that the correct result is reached. In the dynamic approach, however, the optimization function can have local minima with the changes of M. The algorithm must therefore be able to reallocate more than one cluster at a time. This can be major source of inefficiency if not properly designed.

2. Clustering Problem

Clustering aims at partitioning a given set of N data vectors into M groups so that similar data vectors are grouped together and dissimilar data vectors to different groups. We assume that the data set is normalized so that some standard distance metric, e.g. *Euclidean distance*, can be applied. This allows us to estimate the goodness of a solution of M clusters by calculating the *mean square error* (MSE) of the distances from data vectors to their cluster centroids. We also assume that the clusters are spherical with equal variances.

2.1 Algorithm for static clustering

The static clustering problem can be solved reliably using *Randomized Local Search* (*RLS*), which provides simple and effective method for finding the correct location of the clusters [7]. The method is based on trial-and-error approach as follows. New candidate solutions are generated by *random swap* operation, which reallocates a randomly chosen cluster to another part of the vector

space. The new cluster is located to the position of a randomly drawn vector from the data set. The partition is then modified according to the change in the clustering structure, and few iterations of the GLA are applied as fine-tuning. The new solution replaces the previous one only if it decreased the error value. The longer the algorithm is iterated, the better is the clustering.

2.2 Determining the number of clusters

In many cases, the number of clusters is not known beforehand but solving the correct number of clusters is part of the problem. The simplest approach is to generate solutions for all possible number of clusters M in a given range $[M_{min}, M_{max}]$, and then select the best clustering according to a suitable evaluation function f. This approach is referred here as *Brute Force* (BF) algorithm. It allows us to use any static clustering algorithm in the search.

The choice of the evaluation function is a vital part of the clustering; several candidates were presented in [9]. In principle, any function could be used to guide the search; if the evaluation function and the clustering algorithm are properly chosen, BF will find the correct solution but the algorithm will be slow.

In [8], we considered an improved approach that utilizes the best solution for the previous number of clusters m when searching for the solution of m+1 clusters. The algorithm adds one more cluster in the previous solution and applies any iterative algorithm such as the GLA or RLS. The rationale for this is that the previous solution is close to the best solution for current number of clusters and therefore fewer iterations are needed.

3. Dynamic Local Search

We next generalize the RLS method so that it solves the number and the location of the clusters jointly. We refer this algorithm as *Dynamic Local Search* (DLS). The input are the data set (X), an initial solution (C, P), and the search range for the number of clusters (M_{min}, M_{max}). The algorithm applies elementary operations to the current solution and proceeds as the RLS, see Figure 1. There are two main differences to RLS. First is that the only operation is not the random swap, but we may add or remove clusters. The second is that we must use an evaluation function to solve the correct number of clusters.

3.1. Elementary operations

Changing the solution is done in two ways. First way is to use GLA iterations inside DLS to improve the solution towards the closest minimum. The second way is to apply an operation that alters the solution. This change, unlike GLA-iterations, does not necessarily result in better solution in itself, but it allows the search to proceed away from local minimum. We use the following elementary operations for modifying the current solution:

- Cluster swapping,
- Cluster addition,
- Cluster removal.

```
DLS(X, C, P, M_min, M_max) return C, P
C, P ← RandomSolution(X, M_min);
FOR all i ∈ [1, N] DO p_i ← j such that x_i is nearest to c_j;
FOR a ← 1 TO NumberOfIterations DO
    C_new ← Operation(C, M_min, M_max);
    C_new, P_new ← GLA(X, C_new, M);
    IF f(X, C_new, P_new) < f(X, C, P) THEN
        C ← C_new;
        P ← P_new;
END FOR
Return C, P;
```

Figure 1. Pseudocode for the dynamic local search.

The cluster swapping operation is the same as random swap in RLS. Cluster addition creates new clusters by randomly selecting vectors from the data set. Cluster removal deletes randomly chosen clusters centroid from the current solution. Figures 2 and 3 illustrate the addition and removal processes. These operations allow the algorithm to adjust the number of clusters.

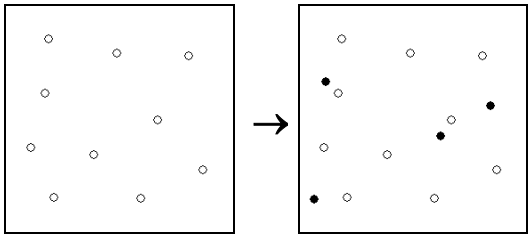

Figure 2. Adding four random centroids.

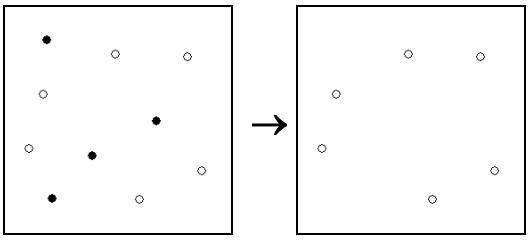

Figure 3. Removing four centroids randomly.

3.2. Amount of change

In the algorithm, in each iteration we first select the operation that is applied. We use the following probabilities: cluster swap 50%, cluster addition 25%, cluster removal 25%. Next we select the number of clusters that are added or removed. In the case of cluster swap, swapping single cluster is enough [7]. In principle, adding or removing one cluster would also be enough to

reach all possible number of clusters. Since the evaluation function may have local minima, bigger changes must take place.

A simple way to control the amount of change is to select the number of clusters to be added or removed (ΔM) randomly so that small changes have higher probability. Let L be the limit (M_{min} or M_{max}), which we may not go beyond. Current number of clusters is M. We get

$$\Delta M = 1 + \left\lfloor r^{\alpha} \cdot |M - L| \right\rfloor \quad (1)$$

where evenly distributed random number r in the range [0, 1[is raised to power α. The case $\alpha = 1$ gives even distribution. For larger values of α, the distribution is skewed towards small values.

3.3. Stopping criterion

There still remains the question of how many iterations we must perform in order to find the correct number of clusters. Obviously, if we iterate long enough we will eventually find the correct solution, but the amount of work varies from one data set to another.

We designed three heuristic stopping criteria for the static clustering in [8]. A criterion referred as *50-50 ratio* halts the search when the improvement for the latter 50% of the iterations divided by the improvement made during the first 50% of the iterations drops below a given threshold value and a given minimum number of iterations has been performed.

4. Test Results

First we use three synthetic two-dimensional data sets with varying numbers of circular clusters. Cluster counts are 20, 35 and 50 for data sets 1, 2 and 3. The graphs of the evaluation function for the data sets are shown in Figure 4. The range [M_{min}, M_{max}] was set to [2, 75].

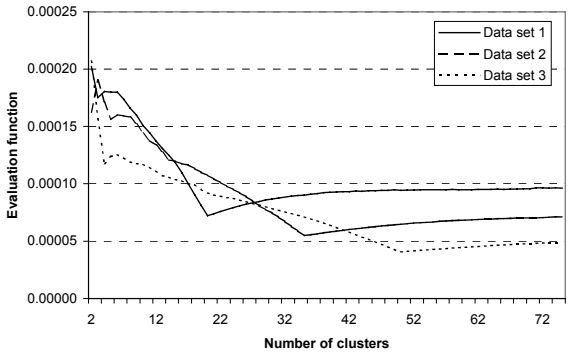

Figure 4. Evaluation function values for the data sets.

4.1. Finding the correct number of clusters

We study how much faster the DLS can find the correct solution than BF search. The general test procedure was to fix the parameters, repeat the algorithm 100 times and calculate how many times the correct number of clusters was found. The number of GLA-iterations used in both BF and DLS was two per iteration.

We ran BF with four different iteration counts for all data sets. The results are shown in Table 1. The total number of iterations clearly shows that while BF will find the correct number it comes at the cost of high total number of iterations. If the range that is searched is wide, then the search will take a long time.

Table 1. Percentage the correct clustering is found by BF.

Iterations: (total)	100 (7400)	200 (14800)	300 (22200)	500 (37000)
Data set 1	83	100	100	100
Data set 2	26	65	83	98
Data set 3	9	22	43	73

We tested the DLS algorithm with 1000 and 2000 iterations using the Eq. (1). Results are shown in Table 2. It can be seen that the Eq. (1) gives good results provided that the α-parameter is set high enough. It is also noted that 1000 iterations is not sufficient for all data sets.

Table 2. Percentage of correct clustering is found by *DLS*.

	1000 iterations				
	α=1.0	α=1.5	α=2.0	α=2.5	α=3.0
Data set 1	80	99	100	100	100
Data set 2	3	37	77	92	98
Data set 3	0	4	26	49	75
	2000 iterations				
	α=1.0	α=1.5	α=2.0	α=2.5	α=3.0
Data set 1	98	100	100	100	100
Data set 2	26	87	99	100	100
Data set 3	2	36	88	95	100

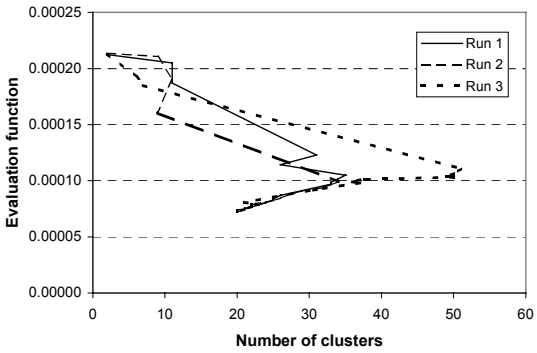

Figure 5. Development of search for data set 1 for three different runs of the DLS algorithm.

Comparison to BF shows that we can find the correct number of clusters with approximately 3% of the amount of work needed by BF. This is significantly better than what was obtained by the *Stepwise* algorithm in [8]; it decreases the number of iterations to about 40%.

Figure 5 shows three examples of how the best solution develops as the search proceeds.

4.2. Stopping criterion

From the previous results we can see that the DLS will find the correct number of clusters when iterated long enough (2000 iterations in the case of test sets used here). It would be useful for the algorithm to stop when the solution has stabilized. We test next whether the stopping criterion introduced in Section 3.3 can be reliably used with DLS. We set the minimal number of iterations to 400, and then apply the static 50-50 ratio with the threshold value of 10^{-5}. Results are shown in table 3.

Table 3. Number of times (%) the correct clustering is found by *DLS* using Equation (1). The numbers in parentheses are the average number of iterations performed.

	α=2.0	α=2.5	α=3.0
Data set 1	98 (861)	98 (670)	99 (626)
Data set 2	93 (1932)	94 (1422)	90 (1295)
Data set 3	91 (3479)	89 (2488)	98 (1906)

We see that the *static 50-50 ratio* can be reliably used in the case of the first data set. In the case of the other two data sets, the results are slightly worse although the algorithm keeps iterating longer. It seems that the static 50-50 ratio works quite well but the optimal choice for the parameters is not trivial.

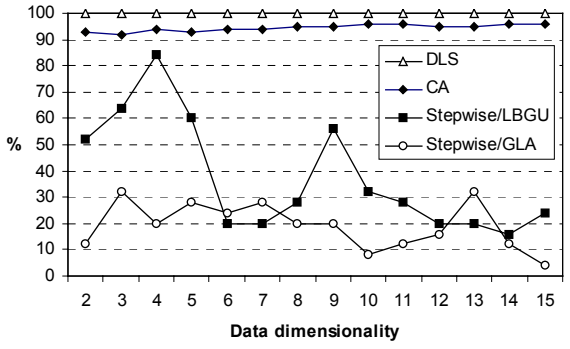

Figure 6. Percentage of finding the correct clustering with different algorithms as function of the dimensionality.

4.3. Comparison with other approaches

We next compare the proposed DLS algorithm with the Competitive agglomeration (CA) [10], GLA [3] LBG-U [11] when integrated into the Stepwise approach as in [8]. The results are summarized in Fig. 6. The main observation is that DLS clearly outperforms the other approaches, of which CA works also reasonably well for these data sets, which are similar to those used earlier except the dimensionality varies and there are 9 clusters.

5. Conclusions

Dynamic local search algorithm was proposed to solve efficiently clustering problems with unknown number of clusters. The algorithm seeks both the correct number of clusters and the optimal positions for the centroids. The algorithm is much faster than simple brute force search.

References

[1] Everitt B.S., *Cluster Analysis*, 3rd Edition. Edward Arnold / Halsted Press, London, 1992.

[2] R. Dubes and A. Jain, *Algorithms that Cluster Data*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[3] Y. Linde, A. Buzo, R.M. Gray, "An algorithm for vector quantizer design", IEEE Transactions on Communications 1980; 28(1): 84-95.

[4] Zeger K, Gersho A, 1989. Stochastic relaxation algorithm for improved vector quantiser design. *Electronics Letters* 25(14), 896-898.

[5] Rose K, Gurewitz E, Fox G, 1990. A deterministic annealing approach to clustering. *Pattern Recognition Letters* 11, 589-594.

[6] Fränti P, Kivijärvi J, Kaukoranta T, Nevalainen O, 1997. Genetic algorithms for large scale clustering problems. *The Computer Journal* 40(9), 547-554.

[7] P. Fränti and J. Kivijärvi, "Randomized local search algorithm for the clustering problem", Pattern Analysis and Applications 2000; 3(4): 358-369.

[8] I. Kärkkäinen and P. Fränti, "Stepwise clustering algorithm for unknown number of clusters", Univ. of Joensuu, Dept. of CS, Report A-2002-5. (submitted)

[9] Bezdek JC, Pal NR, 1998. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man and Cybernetics* 28(3): 302-315.

[10] Frigui H, Krishnapuram R, Clustering by Competitive Agglomeration. Pattern Recognition 1997; 30(7): 1109-1119.

[11] Fritzke B, The LBG-U method for vector quantization – an improvement over LBG inspired from neural networks. Neural Processing Letters 1997; 5(1): 35-45.