STEPWISE ALGORITHM FOR FINDING UNKNOWN NUMBER OF CLUSTERS

Ismo Kärkkäinen and Pasi Fränti

Ismo.Karkkainen@cs.joensuu.fi, Pasi.Franti@cs.joensuu.fi
University of Joensuu, Department of Computer Science, P.O. Box 111, FIN-80101 Joensuu, FINLAND

Abstract

We consider a criterion-based approach to solve dynamic clustering problems. We propose two novel improvements for reducing the work load to find the correct clustering. The first idea is to use stepwise clustering algorithm that utilizes the previous solution when solving the next clustering with different (one more or one less) number of clusters. The second idea is to use a heuristic stopping criterion in the algorithm that solves a single clustering. The stopping criterion estimates potential further improvement on the basis of the improvement achieved so far. The iteration will automatically stop if the estimated improvement stays below a predefined threshold value. By experiments we have found out that any threshold value 0.1 % or less results in the correct clustering with a confidence of more than 99%. Their effect on the run time and clustering quality is studied.

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, 3]. Clustering aims at solving two main problems: *how many clusters* there are in the data set and *where* they are located. We define 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. Clustering is also needed for designing a *codebook* in vector quantization [4].

Static clustering problem can be solved by methods such as *Generalized Lloyd algorithm* (GLA) [5], *simulated annealing* [6], *deterministic annealing* [7,8], *genetic algorithm* [9, 10], *agglomerative methods* [11], and *local search* [12, 13] among many others. The *Randomized Local Search* (*RLS*) [12] is a simple and effective method for solving the problem iteratively. It is based on trial-and-error approach, where the location of the clusters are tentatively changed using *random swapping*. Thus, it corrects the location of misplaced cluster one by one. The ease of implementation makes it possible to tailor the RLS algorithm for various clustering applications with different distance metrics and evaluation criteria. For example, *stochastic complexity* [14, 15] has been successfully applied with the RLS method as the evaluation function for the classification of bacteria in [16].

Dynamic clustering problem can be solved using heuristic methods to determine the number of clusters. For example, *competitive agglomeration* [17] decreases the number of clusters until there are no clusters smaller than a predefined threshold value. The drawback is that the threshold value must be experimentally determined. Divisive clustering, such as the *X-means* in [18], uses an opposite, top-down approach for generating the clustering. The method starts with a single cluster, and new clusters are then created by dividing existing clusters. The splitting continues until a predefined stopping criterion is met. The divisive approach typically requires much less computation than the agglomerative clustering methods but are far less used because of inferior clustering results.

Criterion-based approach, on the other hand, divides the problem in two parts. First a suitable *evaluation function* is defined, which includes the number of clusters as parameter. Static clustering problems are then solved for all reasonable numbers of clusters by using any existing clustering algorithm. The resulting solutions are compared using the evaluation function, and the final result is the clustering that minimizes the evaluation function. Criteria such as *Davies-Bouldin index* [19], *variance-ratio F-test* [20], *stochastic complexity* [15], or *minimum description length* [21] can be applied as the evaluation function among many others [22]. The advantage of the criterion-based approach is that the existing solutions for the static clustering can be utilized as such.

In this paper, we consider the criterion-based approach, and generalize the RLS method to the dynamic clustering problem where the number of clusters is unknown. We propose two variants: (1) *brute force algorithm*, and (2) *stepwise algorithm*. The brute force algorithm applies the RLS method independently to every reasonable number of clusters. It guarantees that the optimal solution will be found if the evaluation criterion is properly designed, and if the RLS method is iterated sufficiently long.

The brute force algorithm is general but not very efficient. For example, we can solve static clustering problem for a data set with 5000 data vectors and 32 clusters in about 40-50 seconds on a 500 MHz Pentium-III machine. However, to solve the dynamic clustering problem when the number of clusters is varied from 1 to 50, the total run time would be about 40 minutes. We propose two methods to speed-up the brute force method.

The first speed-up idea is the stepwise algorithm. It is based on the idea that if we have already found a solution for *m* clusters, then this solution can be utilized when searching for the solution of *m* + 1 clusters. It is expected that these two solutions do not differ greatly from each other and, therefore, fewer iterations would be needed. The proposed method starts by finding solution for *m* = 2, and then repeating the process until a predefined number $m=M_{\max}$. At each step, one new cluster is added and the RLS algorithm is then applied. In principle, any other iterative clustering algorithm could also be applied within the method. We consider three possible algorithms: the GLA, LBG-U, and the RLS.

The second speed-up idea idea is to use a heuristic stopping criterion instead of a fixed number of iterations in the RLS algorithm. The stopping criterion estimates the potential further improvement on the basis of the improvement achieved so far. The iteration will stop if the estimated improvement stays below a predefined threshold value (e.g. 0.1 %). With the brute force algorithm, the RLS is iterated about 179-209 times, on average, and with the stepwise method only about 64-84 iterations. More importantly, the method removes the need for manual tuning of the number of iterations.

2. STATIC CLUSTERING PROBLEM

Static clustering problem can be defined as follows. Given a data set *X* of *N* vectors x_i, partition the data set into *M* disjoint classes so that similar vectors are grouped together and dissimilar vectors belong to different classes. Partition *P* defines the classification by giving each vector an index p_i of the class to which the vector is assigned to. Each class is represented by its representative vector c_j, which is here defined as the centroid of the cluster.

We assume that the vectors of the data set are normalized and that they are in metric space, so that we can use Euclidean distance to estimate the distances between the vectors. This allows us to estimate the goodness of solution of *M* clusters by calculating the distances from vectors to their cluster centroids. We also assume that all clusters are spherical, so all we need to do is to measure the distance to the centroid when calculating the distance to a cluster.

A well-known clustering algorithm for minimizing MSE is the Generalized Lloyd algorithm [6], also known as the LBG. It is simple to implement and can be applied to any initial solution. It proceeds by changing the representation between centroid and mapping descriptions of the clustering using two optimization steps until the solution does not improve anymore. It has been shown that the algorithm converges to a local or global optimum.

Another virtually as simple clustering method is the randomized local search (RLS), which has been shown to outperform most of the comparative algorithms [12]. The RLS proceeds from a given initial solution by replacing a randomly selected centroid with a random data vector, and then fine-tuning the solution using the GLA. The initial solution is a random set of centroids. Our experiments indicate that it is better to apply only a few of GLA-iterations than to allow the GLA to converge to the local optimum [23]. Pseudocode for the RLS is presented in Fig. 1.

```
RLS(X, C, P, M) return C, P
FOR all $i \leftarrow [1, N]$ DO
        $p_i \leftarrow \arg\min_{1 \le k \le M} d(x_i, c_k)$;
FOR a $\leftarrow$ 1 TO NumberOfIterations DO
        $C^{new} \leftarrow C$;
        $j \leftarrow random(1, M)$;  $i \leftarrow random(1, M)$;
        $c_j^{new} \leftarrow x_i$;
        $C^{new}, P^{new} \leftarrow GLA(X, C^{new}, M)$;
        IF $MSE(X, C^{new}, P^{new}) < MSE(X, C, P)$
        THEN $C \leftarrow C^{new}$;  $P \leftarrow P^{new}$;
END FOR
Return C, P;
```

Fig. 1: Pseudocode for randomized local search.

3. DYNAMIC CLUSTERING PROBLEM

We consider the dynamic clustering problem, where also the number of clusters must be solved. We recall first two suitable criteria for evaluating the clustering in Section 3.1. The *brute force algorithm* (*BF*) for finding the best clustering is then introduced in Section 3.2, and the *stepwise clustering algorithm* (*Step*) in Section 3.3. Three stopping criterion are then proposed in Section 3.4.

3.1. Evaluation criteria

MSE has the flaw that it does not take into account the number of clusters. It has the property that the greater the number of clusters is, the lower values we get; the limit being the case where the number of clusters equals the number of data vectors and the MSE-value will be zero. Therefore, we cannot use MSE to determine the number of clusters.

We consider two criteria: *Davies-Bouldin index* (Davies and Bouldin [19]), which has been used in [22, 23, 24], and *variance-ratio F-test* [20]. Davies-Bouldin index (DBI) measures for each cluster the ratio of the intracluster distortion relative to the inter cluster distance of the nearest cluster. This is denoted here as the *mixture* of the two clusters *j* and *k*, and is calculated as:

$$R_{j,k} = \frac{MAE_j + MAE_k}{d(c_j, c_k)} \tag{1}$$

Here *d* is the distance between the cluster centroids, and MAE_j and MAE_k are the mean absolute errors within the clusters *j* and *k*. The higher the intracluster distortion and the closer their centroids, the higher is the index *R*.

The *mixture* of a cluster *j* is defined as the maximum mixture between cluster *j* and all other clusters. The overall DBI-value is then calculated as the average mixtures of the clusters:

$$DBI = \frac{1}{M}\sum_{j=1}^{M} \max_{j \neq k} R_{j,k} \tag{2}$$

The second criterion we use is based on a statistical ANOVA test procedure [20], which we have modified slightly. We omit checking the obtained values against *F*-distribution and use the values directly instead. We refer this criterion as *F-test*.

In brief, the idea is to evaluate all potential clusterings with an evaluation criteria, and determine the one with the minimum value. This procedure is demonstrated for the data set 3 (see Section 4) using the F-test in Fig. 2 as a function of the number of clusters. In this example, there is clear downward peak in the graph. It is noted, that the individual clustering obtained might be suboptimal and, therefore, it is not guaranteed that the method will always find the correct number of clusters.

3.2. The Brute Force (BF) algorithm

The simplest search strategy is to go through a specified range of number of clusters, and find the best solution for every possible number. The solutions are then evaluated and the one that minimizes the chosen evaluation function is the final result of the clustering. This search strategy can use any clustering algorithm to find the individual solutions but we will use the RLS due to its benefits discussed in Section 2. The algorithm is expected to find the correct number of clusters, provided that the chosen evaluation function can distinguish it. The main drawback of the algorithm is that it is time-consuming to generate all possible clusterings in the range [$M_{\min}$, $M_{\max}$].

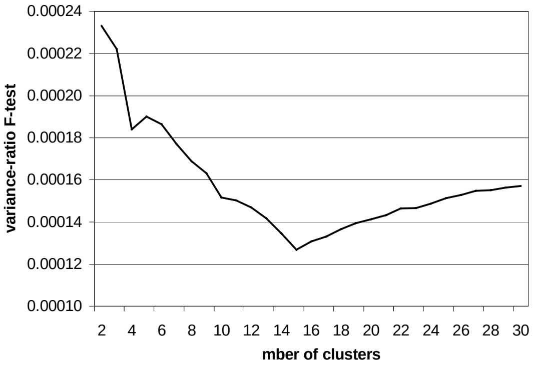


Fig. 2: Values of variance-ratio F-test as a function of the number of clusters.

3.3. Stepwise algorithm

We next describe how to speed up the brute force algorithm without sacrificing its thoroughness. Since we can freely decide the initial solution that is passed on to the RLS, we can utilize the previous solution. If, for example, we have gained a good solution with *m* clusters, then this solution can be used as a starting point for finding the solution with *m*+1 clusters. We perform this by adding a new cluster centroid, performing repartitioning of the data vectors, and then starting the RLS iterations from this solution.

We expect that the clusters in the solution with *m*+1 clusters are located mainly in the same places as in the solution with *m* clusters. This is demonstrated in Fig. 3 as a sequence of clusterings with different number of groups. Cluster centroids are shown as white circles. The clustering results are similar for the three middlemost images. Thus, the hypothesis is that fewer number of iterations are needed for optimizing the solution because most of the clusters are already in their correct location.

The stepwise algorithm involves two main design questions: whether we should add or remove clusters, and how the clusters are modified. The first design question can be divided into two approaches. In the first approach (denoted as *Step*$^+$), we start with a solution of $M_{\min}$ clusters

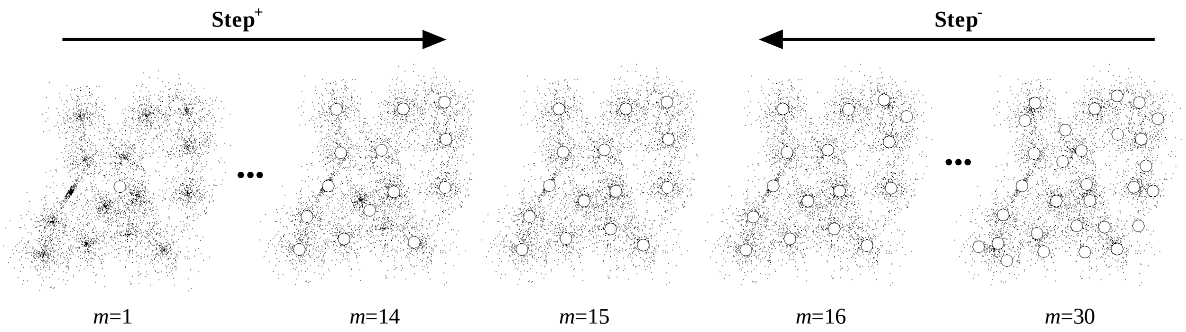


Fig. 3: Location of different number centroids (white dots) in the data.

and add new clusters one at a time in the main loop. In the second approach (denoted as *Step*$^-$), we start with a solution of $M_{\max}$ clusters and remove the clusters one at a time. With brute force algorithm, it does not matter whether we increase or decrease the number of clusters in the loop; the result will be the same as the solutions are generated independently from each other. In the stepwise approach, however, the direction can have an effect on the expected number of iterations needed.

The second design question is to define how we add or remove the clusters. We derive the method directly from the original RLS algorithm and perform the change randomly. This is argued by the fact that the correct place of the clusters must be optimized by a series of random swap operations anyway. Thus, the *Step*$^+$ algorithm inserts new cluster into a random location. The *Step*$^-$ algorithm removes a randomly chosen cluster. Pseudocode for the *Step*$^-$ algorithm is shown in Figure 4.

```
Step⁻($X$, $M_{min}$, $M_{max}$) return $C$, $P$

$C$, $P \leftarrow$ RandomSolution($X$, $M_{max}$);
$C_{best}$, $P_{best} \leftarrow$ RLS($X$, $C$, $P$, $M_{max}$);
FOR $m \leftarrow M_{max} - 1$ DOWNTO $M_{min}$ DO
    $C \leftarrow$ Remove random centroid
    $C$,$P \leftarrow$ RLS($X$, $C$, $P$, $m$);
    IF $f(X, C, P) < f(X, C_{best}, P_{best})$ THEN
        $C_{best} \leftarrow C$; $P_{best} \leftarrow P$;
END FOR
return $C_{best}$, $P_{best}$;
```

Fig. 4: Stepwise algorithm Step$^-$ for decreasing number of clusters.

3.4. Stopping criterion

In general, it is not clear how many iterations we need to perform the RLS in order to find the correct clustering. This question arises both in the brute force and in the stepwise-algorithms. In the stepwise-algorithm the question is even more important because the aim is to decrease the number of iterations and yet obtain the correct number of clusters. We consider next few simple heuristics that can be used for estimating whether it is still worthwhile to continue improving the solution by the RLS. If we had an algorithm that always improves the solution in each iteration this would not be needed as we could check against the previous value and stop if difference were small enough. Since this is not the case (most likely the best solution remains the same) we can take this into account in the first criterion.

We call the first heuristic *T-times*, as it stops after the solution has been iterated T times since the last improvement to solution occurred. For example, if T is 3 and the previous improvement occurred in the iteration number 100, we will stop in iteration 300 if there will appear no further improvement. We also need a minimum number of iterations that must be performed so that we do not stop right at the beginning. Let f_i be the value of the solution after iteration i, i.e. the f-value for the currently best C and P, and let k be the current iteration number. Stopping condition is:

$$k > T_{\min} \ \wedge \ \left(f_{k/T} - f_k = 0\right) \tag{3}$$

Second heuristic is called as *50-50 ratio*, as it stops when the improvement for the latter half ($f_{k/2}$ - f_k) divided by the improvement that occurred in the first half (f_1 - $f_{k/2}$) falls below certain limit. Taking first and second halves is a consequence of the fact that the solution doesn't improve in every iteration. The following formula gives the appearance of smoothly decreasing error:

$$k > T_{\min} \ \wedge \ L > \frac{f_{k/2} - f_k}{f_1 - f_{k/2}} \tag{4}$$

Here L is the predefined threshold value. As with the first heuristic, we need to specify the minimum number of iterations, otherwise we might stop immediately. Fig. 5 shows howthe values relate to the number of iterations.

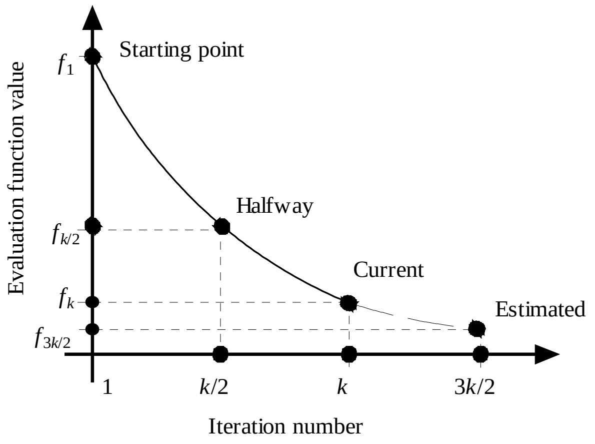


Fig. 5: Values used with stopping criteria.

Third heuristic is called as *Estimate*, as it tries to estimate the future improvement of f relative to the improvement so far. The estimated value is shown in Fig. 5 as $f_{3k/2}$. We calculate the estimated value by assuming that the relative slowdown of the improvement of the solution from k iterations to 3k/2 iterations when compared to the slowdown from k/2 iterations to k iterations is the same as that for ranges from k/2 to k and 1 to k/2. The estimated value is:

$$f_{3k/2} = \left(f_{k/2} - f_k\right) \cdot \frac{f_{k/2} - f_k}{f_1 - f_{k/2}} = \frac{\left(f_{k/2} - f_k\right)^2}{f_1 - f_{k/2}} \tag{5}$$

We calculate the ratio of the estimated future improvement and the total improvement ($f_1 - f_k$) in order to make the heuristic independent of the range of values of the criterion function f. We must perform some minimum number of iterations in order not to stop right away. The heuristic is:

$$k > T_{\min} \ \wedge \ L > \frac{f_{3k/2}}{\left(f_1 - f_k\right)} \tag{6}$$

The numerator estimates how much the solution would improve if the number of iterations would increase by half, presuming that the rate of improvement decreases at the same rate as it decreased between first half and second half. The denominator normalizes the value so that we get the relative improvement estimation in respect to the total improvement obtained so far. Thus we do not need to take into account the range of values of the evaluation function.

4. TEST RESULTS

We use the four data sets shown in Fig. 6 with varying complexity in terms of spatial data distributions. The test sets have 5000 data vectors randomly scattered around 15 predefined clusters with a varying degree of overlap between the clusters.

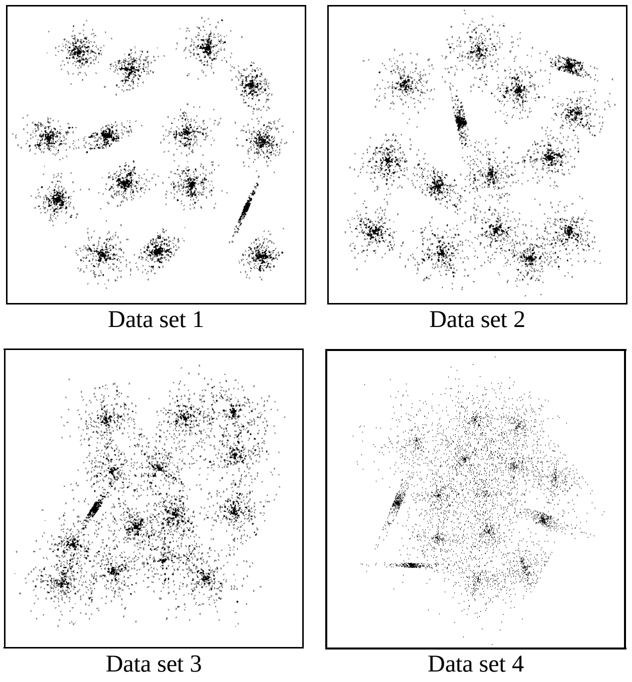


Fig. 6: Two-dimensional data sets.

We first study the parameter setup of the proposed method by finding out how many iterations the proposed algorithms need to be run in order to find out the correct number of clusters. This describes the minimum amount of work that must be performed. We perform clustering for all four data sets using the different clustering algorithms, evaluation and stopping criteria.

4.1. Fixed number of iterations

In order to estimate the minimum work load, we first study the work load required by the brute force algorithm (BF) when manually fixing the number of iterations.

The results are summarized in Figures 7 and 8 for the data set 3 by showing the percentage the algorithm found out the correct number of clusters. In the case of F-test, the brute force algorithm requires approximately 100 iterations to consistently reach the correct answer (Fig. 7).

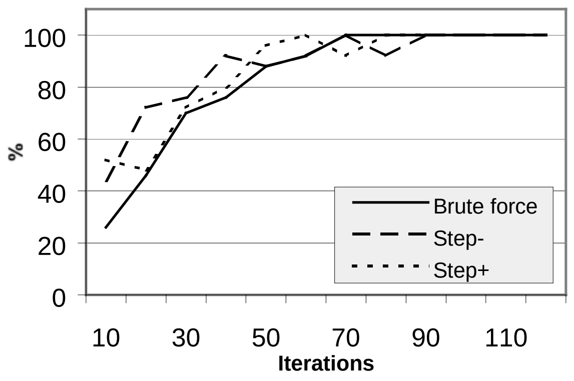
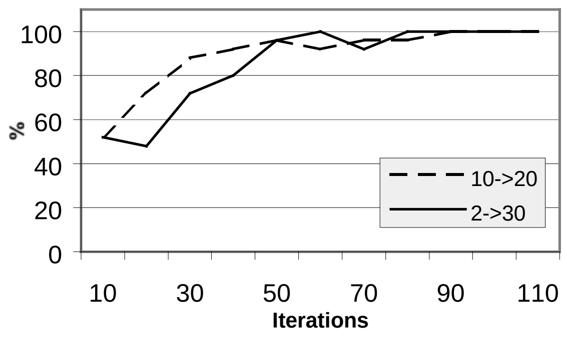


Fig. 7: The percentage the algorithms found out the correct answer, as a function of the number of iterations.

The stepwise algorithm does not start from scratch as BF but it uses the previous solution as a starting point. The quality of the result may therefore depend on the search range of the number of clusters applied. This might have an effect on the quality of the result in case if only a small number of iterations is applied, and when the search range is limited. However, Fig. 8 indicates that the results are virtually the same regardless of the range and direction of the search. For example, the choice between Step$^+$ and Step$^-$ do not have any significant effect on the result. To sum up, there are no significant differences between the three algorithms (BF, Step$^+$ and Step$^-$) when a fixed number of iterations were applied.

4.2. Using stopping criteria with brute force

In the previous chapter we studied the minimum work load required by the different algorithms. In practice, the correct clustering is not known beforehand and the number of iterations cannot be manually tuned. The use of a proper stopping criterion is therefore important for minimizing the work load. The aim is to get the correct clustering with the smallest number of iterations as possible. The results in Table 1 can be used for estimating the lower limit of this work load.



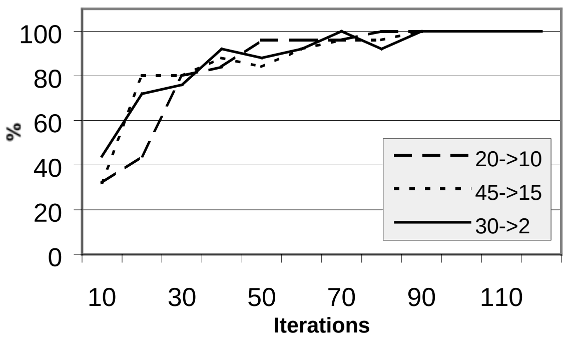


Fig. 8: The percentage the algorithms found out the correct answer, as a function of the number of iterations. The results are for the Step$^+$ (top) and Step$^-$ (bottom) algorithms with different search ranges using F-test with data set 3.

Table 1: The minimum number of iterations required to find out the correct answer with 100% of the cases.

	F-test			DBI		
	BF	Step$^+$	Step$^-$	BF	Step$^+$	Step$^-$
Data set 1	130	60	60	90	100	70
Data set 2	70	60	70	1200	900	800
Data set 3	70	80	70	N/A	N/A	N/A

The brute force and stepwise algorithms were tested using the three stopping criteria presented in Section 3.4 (*T-times*, *50-50 ratio*, *Estimate*). The threshold parameter was experimentally set to *T*=3 for *T-times*, and *L*=0.001 for the other two criteria. These criteria require also some minimum number of iterations before the stopping criterion is allowed to take effect. We studied the performance of the criterion by varying the minimum number of iterations from 10 to 90. The results are summarized in Table 2 as the average number iterations applied with the first parameter setup that were able to produce the correct clustering in all runs.

The results show that the number of iterations remains reasonably low when *F-test* is used as the evaluation function. For example, the BF was iterated only 122-162 times (on average) in the case of the *50-50 ratio test*, and 179-209 times in the case of the *Estimate* in order to get the correct clustering. The corresponding numbers for the Stepwise are 88-183 (with *50-50 ratio test*) and 81-121 (with *Estimate*). The average number of iterations behaves basically in the same way with the stepwise algorithms as with the brute force algorithm. The *T-times* performed less consistent in these tests as it keeps iterating unnecessarily long with the more difficult test sets (Set 3 and 4).

With *DBI* the results are not as good as with the *F-test*. The correct results is found only in the case of the data set 2, and even then, the algorithm sometimes iterates unnecessarily long. The average value of the stepwise algorithms is affected by a single large iteration count, where for 28 clusters the RLS algorithm performed 244,053 iterations. As the threshold value was set to *T*=3, it means that the last improvement has occurred at the 81,351 iteration, after which there has been no improvement whatsoever. In the case of the data set 3, on the other hand, the methods were unable to keep the algorithm iterating long enough in order to find the correct number of clusters for the data set 3.

Table 2: Average number of iterations applied when using a stopping criterion (for data set 3).

	T-times	*50-50 ratio*	*Estimate*
Brute force	881	122	209
Step$^+$	282	88	92
Step$^-$	497	1078	65

4.3. Behavior of the stopping criteria

Despite the fact that the minimum number of iterations in Table 2 may be half of what is required in comparison to Table 1, the average number of iterations may yet be much higher. The stopping criteria try to determine when the solution has ceased to improve, not when enough work has been done to find the correct number of clusters. Another reason is that not all cluster counts require the same amount of iterations. The less there are clusters, the less iterations are needed. After the correct number of clusters has been passed, the number of iterations increases noticeably. This is illustrated in Fig. 9. For *T-times* there is a general rising trend after the correct number of clusters has been exceeded. Similar trend can also be observed for *Estimate*.

The graphs in Fig. 9 show marked jaggedness. This is due to the random nature of the underlying algorithm. Since the RLS does not necessarily improve the solution in every iteration, it is possible that a noticeable drop happens just before the stopping criterion would stop the iterations and the limit is increased. With *T-times* this means that the number of iterations can grow to be extremely large.

Regardless of the stopping criterion used, the differences in average values of the error functions are small despite the number of iterations used. In all cases the solutions are equal to three significant digits. We can therefore conclude that finding the optimal solution and the correct number of clusters are practically the same thing. In other words, if we iterate long enough to find the correct number of clusters, we have also found the optimal solution. Therefore, further iteration for the solution with the correct number of clusters does not seem to be necessary.

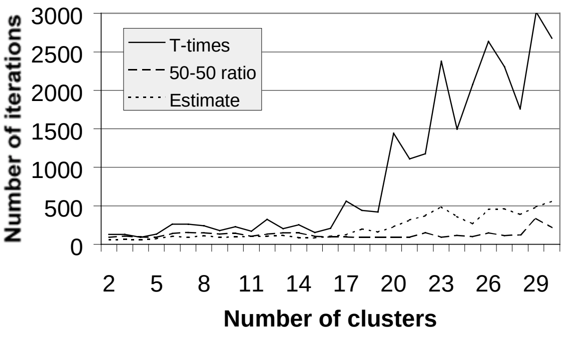


Fig. 9: Average number of iterations as function of the number of clusters.

4.4. Comparison with the existing algorithms

We next compare the following approaches:

- Stepwise with GLA
- Stepwise with LBG-U
- Stepwise with RLS (the proposed method)
- Competitive agglomeration (CA)
- Brute Force (BF)

The Stepwise GLA combines the *Step-* approach with the GLA [5] as follows. It generates M_{max}=30 random clusters, and then proceeds to decrease the number of clusters by removing each cluster in turn. There are M different choices for the cluster removal, and every of them are iterated by the GLA until convergence. As a result, we obtain a set of candidate solutions, of which we keep the best one for the initial solution for next cluster removal.

The Stepwise with LBG-U is implemented in the same manner as the Stepwise GLA but the LBG-U algorithm [13] is used instead. The LBG-U is also similar to that of the RLS but it uses deterministic choice for replacing the location of the cluster, whereas the RLS uses random choice. This also means that we do not need to repeat the RLS for every possible cluster removal but we can simply start from random starting point as the algorithm will anyway optimize the cluster locations. With the GLA and LBG-U, however, the repeats are done as there are no randomness in this part and all possibilities can be explored.

The competitive agglomeration [17] uses competitive learning for optimizing the location of the clusters. The algorithm starts with a predefined number of initial clusters, and then removes all clusters that are smaller than a predefined threshold value ε. The process continues until there are no more clusters to be removed and the solution has converged to optimum. The initial number of clusters is set to 100, and the threshold value was experimentally determined by varying it from 10^{-6} to 10^{-1}.

Table 3: The number of times the correct clustering was found (among 100 repeats) by the three Stepwise variants, Brute force, and by Competitive agglomeration.

	DBI				
	Stepwise GLA	Stepwise LBG-U	Stepwise RLS	Brute Force	CA
Data set 1	14 %	94 %	98 %	95 %	70 %
Data set 2	19 %	53 %	96 %	100 %	8 %
Data set 3	28 %	72 %	76 %	82 %	0 %
Data set 4	31 %	36 %	33 %	52 %	0 %
Iris data	0 %	0 %	0 %	0 %	5 %

	F-test			
	Stepwise GLA	Stepwise LBG-U	Stepwise RLS	Brute Force
Data set 1	22 %	94 %	98 %	98 %
Data set 2	30 %	74 %	100 %	98 %
Data set 3	39 %	80 %	100 %	100 %
Data set 4	51 %	92 %	100 %	100 %
Iris data	100 %	100 %	100 %	100 %

The results for the data sets 1 to 4, and for the Iris data set [25] are summarized in Table 3. When F-test is used as the criterion, the proposed method (Stepwise RLS) finds the correct clustering almost in all cases. The Stepwise with LBG-U works fine most of the times but the Stepwise GLA gives significantly worse success rate with all data sets. When DBI is used, the results are worse with all variants. The relative performance of the different methods is similar to that of the F-test. The CA, on the other hand, fails to find the correct clustering except in the case of the more difficult data sets. It tends to remove too many clusters no matter of the parameter setup. The results in Fig. 10 show that, besides the LBG-U, the algorithms performance is consistent on the change in the dimensionality.

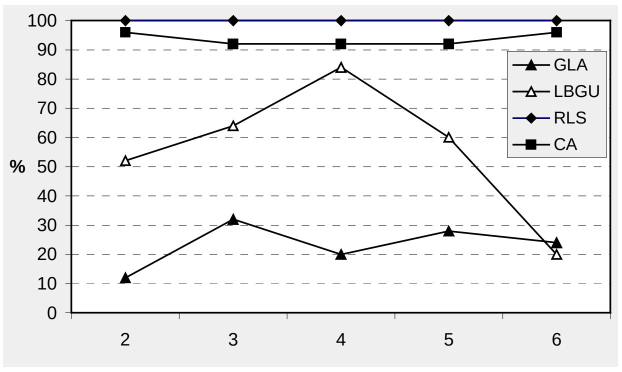


Fig. 10: Comparison of the algorithms as a function of the data set with varying dimension (from 2 to 6).

5. CONCLUSIONS

We have introduced stepwise clustering algorithm for finding the correct clustering in a case when the number of groups is unknown. The method generates solutions for all cluster counts within a given search range.

According to the experiments, the work load can be reduced down to about 200 iterations per cluster count with the Brute Force approach. With the stepwise algorithm, the algorithm iterates only about 65-85 times on average. The results indicate that there is obvious dependency between the number of iterations and the spatial complexity of the data set (clusters overlap). The choice of the evaluation function seems also to be important.

6. REFERENCES

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

[2] Kaufman L, Rousseeuw PJ, Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley Sons, New York, 1990.

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

[4] Reeves C, Modern Heuristic Techniques for Combinatorical Optimization Problems. McGraw - Hill, 1995.

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

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

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

[8] Hoffmann T, Buhmann J, Pairwise Data Clustering by Deterministic Annealing. IEEE Transactions on Pattern Analysis and Machine Intelligence 1997; 19(1): 1-14

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

[10] Fränti P, Genetic algorithm with deterministic crossover for vector quantization. Pattern Recognition Letters 2000; 21(1): 61-68.

[11] Ward JH, Hierarchical grouping to optimize an objective function. J. Amer. Statist.Assoc. 1963; 58: 236-244.

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

[13] 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.

[14] Rissanen J, Stochastic Complexity. Journal of Royal Statistical Society B 1987; 49(3): 223-239.

[15] Gyllenberg M, Koski T, Verlaan M, Classification of binary vectors by stochastic complexity. Journal of Multivariate Analysis 1997; 63(1): 47-72.

[16] Fränti P, Gyllenberg HH, Gyllenberg M, Kivijärvi J, Koski T, Lund T, Nevalainen O, Minimizing stochastic complexity using local search and GLA and with applications to classification of bacteria. Biosystems 2000; 57(1): 37-48.

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

[18] Pelleg D, Moore A, X-means: Extending K-means with Efficient estimation of the Number of Clusters. Proc. 17th International Conf. on Machine Learning. Morgan Kaufmann, San Francisco CA 2000; pp. 727-734.

[19] Davies DL, Bouldin DW, A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1979; 1(2): 224-227.

[20] Ito PK, Robustness of ANOVA and MANOVA Test Procedures. In: Krishnaiah PR (ed). Handbook of Statistics 1: Analysis of Variance. North-Holland Publishing Company, 1980, pp 199-236.

[21] Bischof H, Leonardis A, Selb A, MDL Principle for robust vector quantization. Pattern Analysis and Applications 1999; 2(1): 59-72.

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

[23] Kärkkäinen I, Fränti P, Minimization of the value of Davies-Bouldin index. In Proceedings of the IASTED International Conference on Signal Processing and Communications (SPC'2000). IASTED/ACTA Press, 2000, pp 426-432.

[24] Sarkar M, Yegnanarayana B, Khemani D, A clustering algorithm using an evolutionary programming-based approach. Pattern Recognition Letters 1997; 18(10): 975-986.

[25] Duda RO, Hart PE, Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons, New York, page 218, 1973.