

Improvements to the scalability of multiobjective clustering

Julia Handl

University of Manchester
J.Handl@postgrad.manchester.ac.uk

Joshua Knowles

University of Manchester
J.Knowles@manchester.ac.uk

Abstract- In previous work, we have proposed a novel approach to data clustering based on the explicit optimization of a partitioning with respect to two complementary clustering objectives [4, 5, 6]. In a comparison to alternative clustering techniques, the approach showed a high performance in terms of its capability to deal with a range of difficult data properties, including overlapping clusters, elongated cluster shapes and unequally sized clusters. In this paper, we make three modifications to the algorithm that improve its scalability to large data sets with high dimensionality and large numbers of clusters. Specifically, we introduce new initialization and mutation schemes that enable a more efficient exploration of the search space, and modify the null data model that is used as a basis for selecting the most significant solution from the Pareto front. The high performance of the resulting algorithm is demonstrated on a newly developed clustering test suite.

1 Introduction

The inherently multiobjective nature of data clustering, as identified in, e.g., [3, 8], has motivated us in our recent work [4, 5, 6] to devise an explicit multiobjective-optimization approach to this problem. In [4], we began to investigate this idea with an evolutionary algorithm, VIENNA, which used a fixed, user-specified, number of clusters, k . Our subsequent work [5, 6] developed the initial approach significantly, introducing a method for automatically estimating the best solutions from the Pareto front returned — at the same time determining the number of clusters, automatically. In the following we recall the main elements of the latter, briefly.

1.1 Existing algorithm

Our existing multiobjective clustering algorithm MOCK (MultiObjective Clustering with automatic K-determination) is based on the elitist multiobjective evolutionary algorithm, PESA-II, described in detail in [2]. MOCK optimizes two clustering objectives, *overall deviation* and *connectivity*, which reflect two fundamentally different aspects of a good clustering solution: the global concept of *compactness of clusters*, and the more local one of *connectedness of data points*. The definitions of these objectives can be found in [6].

The encoding employed is the locus-based adjacency scheme proposed in [9]. In this graph-based representation, each individual g consists of N genes g_1, \dots, g_N , where N is the size of the clustered data set, and each gene g_i can take allele values j in the range $\{1, \dots, N\}$. Thus, a value

of j assigned to the i th gene, is then interpreted as a link between data items i and j : in the resulting clustering solution they will be in the same cluster. The decoding of this representation requires the identification of all subgraphs, and all items belonging to the same subgraph are assigned to one cluster. This can be done in linear time.

The operators used are standard uniform crossover, and a specialized initialization and mutation scheme. MOCK’s initialization is based on minimum spanning trees (MSTs): for a given data set, the complete MST is computed using Prim’s algorithm. The i th individual of the initial population is then initialized by the MST with the $i - 1$ largest links removed. This has the effect of generating solutions with a range of cluster numbers, all solutions having well-separated clusters.

MOCK’s mutation operator allows data items to be linked to one of their L nearest neighbours only. Hence, $\forall i, g_i \in \{nn_{i1}, \dots, nn_{iL}\}$, where nn_{il} denotes the l th nearest neighbour of data item i . This has the effect of significantly reducing the size of the search space.

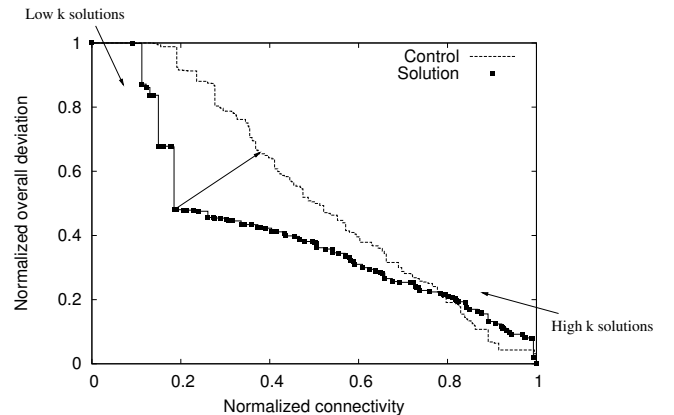


Figure 1: Solution and control reference front for a run of MOCK on a simple four-cluster data set (from [5]). Solutions are scored by their distance to the reference front (their ‘attainment score’). The solution with the largest minimum distance to the reference front is indicated by an arrow, and corresponds to the correct four-cluster solution. In general, all solutions situated at distinct knees in the Pareto front are of high immediate interest. Several such solutions can exist and they often correspond to cluster structures on different levels. In a plot of the attainment scores as a function of the number of clusters, such distinct ‘knees’ are manifest as local optima.

The algorithm generates clustering solutions that correspond to different trade-offs between the two clustering objectives and contain different numbers of clusters. In order to reduce the number of solutions to consider, an automated technique was developed, which selects good solu-

tions from the resulting Pareto front, and, thus, simultaneously determines the number of clusters in a data set (see Figure 1). This method of solution selection is based on the shape of the Pareto front. Specifically, it tries to determine ‘knees’ in the Pareto front that correspond to good solutions. A comparison to a null model, that is, random control data, is used in order to help correctly determine these knees, and abstract from k -specific biases in the two objectives. A detailed motivation and description of this methodology (including pseudo-code) is provided in [5].

In previous work, MOCK has been compared to three single-objective clustering algorithm, an advanced ensemble method [13] and the Gap statistic [14], and results indicated a clear advantage to the multiobjective approach.

1.2 Limitations of MOCK and scope of this work

The first implementation of MOCK described above was targeted at scenarios in which the number of clusters is reasonably small. Specifically, the range of obtainable clusters was restricted to the range [1, 25], and the algorithm was evaluated on data sets with relatively small numbers of clusters (mostly $k \leq 10$). However, a limitation to such small numbers of clusters is not realistic in several real-world applications, in particular in biological applications such as the mining of gene-expression data.

MOCK’s original restriction on the number of clusters is enforced by means of a simple penalty function. While a straightforward adaptation of these penalty functions to allow numbers of clusters $k \geq 25$ is possible, such an extension comes at major costs in the required number of evaluations and/or high impairments in clustering quality. In this paper we therefore suggest three more fundamental modifications that improve the scalability of the algorithm to data sets with large numbers of clusters. The suggested modifications generally result in a more efficient exploration of the search space, improving performance on large and high-dimensional data sets, and significantly reducing the number of evaluations required.

1.3 Paper outline

The remainder of this paper is structured as follows. Section 2 summarizes the modifications introduced to the algorithm. Section 3 describes the generators developed to obtain complex high-dimensional test data. The experimental setup is described in Section 4, while Section 5 presents results. Finally, Section 6 concludes.

2 Modifications

In this section, three modifications are introduced that significantly improve MOCK’s convergence speed and its scalability to data sets with large numbers of clusters.

2.1 Initialization scheme

On the level of *individual genes*, basing MOCK’s initialization scheme on MSTs has the advantage of providing high-quality genotypes, in the sense that data items are linked

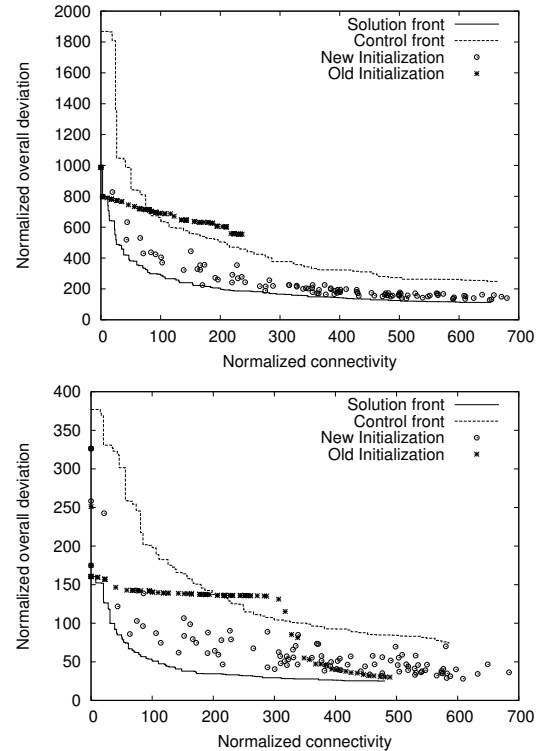


Figure 2: Comparison of the solutions constructed by the original and the new initialization approach on two different data sets (Long1 and Smile1, described in [5]). The additional use of k -means solutions induces a more even distribution of solutions along the Pareto front, in particular towards the bottom right side (where solutions with large numbers of clusters are situated). The use of the criterion of ‘interestingness’ decreases the generation of highly similar MST solutions.

to data items that are nearby and therefore likely to be in the same class. On the *cluster* level, the solutions generated by cutting the longest link in the MST essentially correspond to solutions that would be identified by single link agglomerative clustering. On the one hand, this has the effect that initial solution quality will be very high on data sets with well-separated clusters. On the other hand, this type of solution-construction inherits the shortcomings of single link agglomerative clustering: in the absence of spatially separated clusters the method tends to isolate outliers so that many of the solutions generated are highly similar (this ‘chaining effect’ in decision space carries over to objective space, see Figure 2).

Clearly, a better spread of initial solutions along the Pareto front is desirable, in order to improve MOCK’s convergence to the Pareto front. Towards this end a new approach to solution construction has been developed. The new initialization operator is based on the observation that different clustering algorithms tend to perform better (find better approximations) in different regions of the Pareto front [6]. In particular, MST type or single link solutions tend to be close to optimal in those regions of the Pareto front where connectivity is low, whereas k -means performs well in the regions where overall deviation has significantly decreased. An initialization based on a mixture of k -means

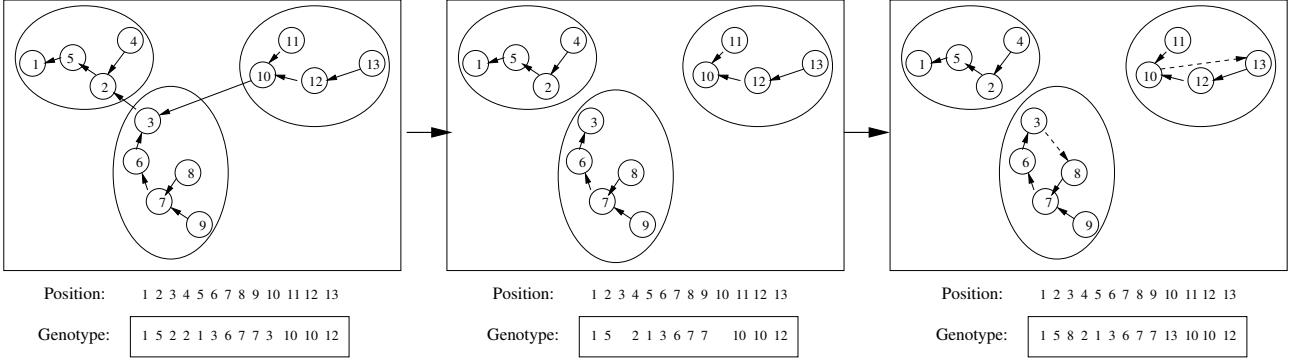


Figure 3: Construction of an MST-similar solution from a given k -means solution (visualized by the ellipses). Starting from the original MST solution, all links that cross the k -means cluster boundaries are removed. The missing links are then replaced by a link to a randomly chosen neighbour with $i = nn_{il} \wedge l \leq L$.

and MST solutions can therefore help to obtain a better spread of solutions and a better initial approximation to the Pareto front.

Generation of interesting MST solutions

First, we consider the generation of MST solutions. In order to avoid the chaining effect mentioned above, we employ a definition of interestingness that distinguishes between ‘uninteresting’ links whose removal leads to the separation of outliers, and ‘interesting’ links whose removal leads to the discovery of real cluster structures.

Definition 1.1: A link $i \rightarrow j$ is considered as *interesting*, iff $i = nn_{ji} \wedge j = nn_{ik} \wedge l > L \wedge k > L$, where L is a parameter. Its *degree of interestingness* is $d = \min(l, k)$.

Definition 1.2: A clustering solution C is considered as *interesting*, if it can be deduced from the full MST through the removal of interesting links only.

For a given data set, a set of interesting MST-derived solutions can then be constructed as follows. In a first step, all I interesting links from the MST are detected and are sorted by their degree of interestingness. Using this sorted list, a set of clustering solutions is then constructed: for $n \in [0, \min(I, 0.5 \times fsize)]$, where $fsize$ is the total number of initial solutions, clustering solution C_n is generated by removing the first n interesting links. The missing links are then replaced by a link to a randomly chosen neighbour j with $j = nn_{il} \wedge l \leq L$.

Generation of k -means solutions

Next, we consider the generation of k -means solutions. We start by running the k -means algorithm (for 10 iterations) for different numbers of clusters $k \in [2, fsize - (\min(I, 0.5 \times fsize) + 1)]$. The resulting partitionings are then converted to MST-based genotypes as illustrated in Figure 3. The preservation of a high degree of MST information (within clusters) at this stage is crucial for the quick convergence of the algorithm. Note that the numbers of

clusters obtained as the final phenotypes are not pre-defined and can increase or decrease depending on the structure of the underlying MST.

2.2 Neighbourhood-biased mutation operator

MOCK’s mutation operator narrows down the search space by restricting the number of data items that an individual item can be linked to. In the original algorithm, each gene has an equal probability $\frac{1}{N}$ of undergoing such a mutation event, per generation.

Intuitively, ‘long links’ are expected to be less favourable, that is, a link $i \rightarrow j$ with $j = nn_{il}$ may be preferred over a link $i \rightarrow j^*$ with $j^* = nn_{ik}$ if $l < k$. This can be used to bias the mutation probability of individual links $i \rightarrow j$, which we now define as

$$p_m = \frac{1}{N} + \left(\frac{l}{N}\right)^2,$$

where $j = nn_{il}$ and N is the size of the data set.

2.3 Null model

The original version of MOCK uses a Poisson model for the generation of the control data, that is, the control data is generated uniformly randomly within the bounds of the given data set (where the bounds are determined as the maximum and minimum value obtained in each dimension). While this approach performs well for low dimensional data, it fails to correctly capture the shape of the data manifold for high-dimensional data (see Figure 4). This problem arises due to the properties of high-dimensional data, which are usually embedded in a lower-dimensional manifold within the high-dimensional space — a property that cannot be captured by the current null model.

Here, we suggest the use of a refined null model, based on the description by Sarle [12]. Again, a Poisson model is used, but now it is set up within the space of principal components. Specifically, a principal component analysis is applied to the correlation matrix of the original data. The eigenvectors and eigenvalues obtained are then used for the definition of a uniform distribution: the data is generated

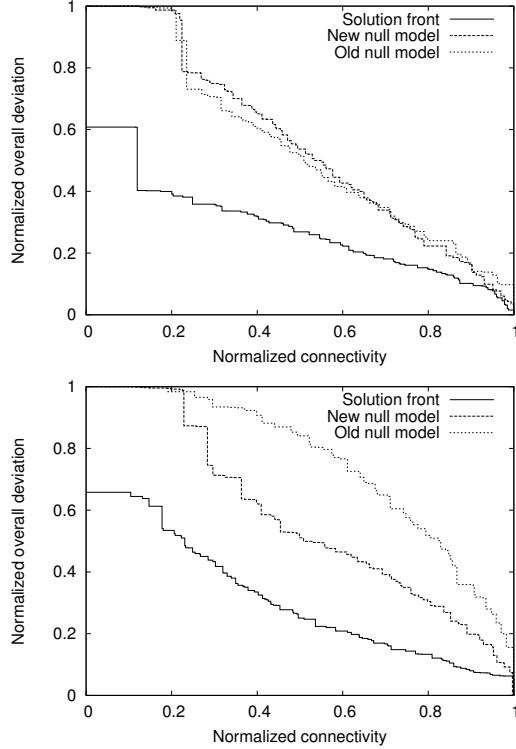


Figure 4: Comparison of the control fronts obtained using the original and the new null model on a two dimensional (top) and a 100-dimensional (bottom) data set, each of which contains four clusters. In low-dimensional space both models yield very similar results. In high-dimensions, the performance of the original null model breaks down and the control fronts generated are of increasingly concave shape.

within a hyperbox in eigenspace, where each side of the hyperbox is proportional in length to the size of the eigenvalue corresponding to this dimension. The resulting data is then back-transformed to the original data space.

3 Data generators

In order to obtain highly complex test data, two new cluster generators were developed. Both generators and the specific test data sets used in this paper are made available at www.dbk.ch.umist.ac.uk/handl/generators/.

3.1 Gaussian cluster generator

The first generator is based on a standard cluster model using multivariate normal distributions. The covariance matrices need to be symmetric and positive definite, which is ensured by constructing them to be diagonally dominant, with positive diagonal elements only.

Briefly, a single multivariate cluster is defined as follows:

1. the mean, uniformly in the range $[-10, 10]$
2. the off-diagonal entries of the covariance matrix, generated as a random number in the range $[-1, 1]$ with a distribution following $\pm y, y = x^2$ where x is a uniformly random deviate in $[0, 1]$ and the sign of y is determined randomly by a pseudorandom ‘coin toss’

Table 1: Gaussian data sets of low dimension. For each of the 8 combinations of cluster number and dimension, 10 different instances were generated, giving 80 data sets in all.

<i>Parameter</i>	<i>range</i>
Number of clusters	4, 10, 20, 40
Dimension	2, 20
Size of each cluster	uniformly in $[50, 500]$ for 4 and 10 cluster instances, and $[10, 100]$ for 20 and 40 cluster instances

3. the diagonal entries of the covariance matrix, generated as the absolute sum of all off-diagonal entries plus a random number in the range $[0, 20 \cdot \sqrt{D}]$ with a distribution following $\pm y, y = x^2$, where x is a uniformly random deviate in $[0, 1]$, the sign of y is determined randomly by a ‘coin toss’, and D is the dimensionality of the data set

Here, the use of a distribution $y = x^2$ serves to encourage the production of elongated clusters.

Data generation is based on a simple trial-and-error scheme. Clusters are iteratively constructed, and a simple heuristic is used to detect overlap between them. Overlapping clusters are rejected and regenerated, until a valid set of clusters has been found. Using this method, 80 different data sets were generated, as described in Table 1.

In low dimensions, the clusters generated are frequently elongated and of arbitrary orientation (see Figure 5). However, in higher dimensions (that is, more than 10), the shape of a cluster becomes more (hyper)spherical and more axis-aligned: the former because a high variance in one dimension hardly affects the Euclidean distance of points when there are very many dimensions; the latter because with higher dimensions the non-diagonal entries in the covariance matrix are forced to be relatively smaller compared with those on the diagonal. Because of the lack of generality of spherical clusters, we have developed a second alternative cluster generator that produces more elongated cluster shapes in arbitrarily high dimensions.

3.2 Ellipsoidal cluster generator

This generator creates ellipsoidal clusters with the major axis at an arbitrary orientation. The boundary of a cluster is defined by four parameters:

1. the origin (which is also the first focus)
2. the interfocal distance, uniformly in the range $[1.0, 3.0]$
3. the orientation of the major axis, uniformly from amongst all orientations
4. the maximum sum of Euclidean distances to the two foci, uniformly in the range $[1.05, 1.15]$ — equivalent to an eccentricity ranging from $[0.870, 0.952]$

Table 2: Ellipsoidal data sets of high dimension. For each of the 8 combinations of cluster number and dimension, 10 different instances were generated, giving 80 data sets in all.

<i>Parameter</i>	<i>range</i>
Number of clusters	4, 10, 20, 40
Dimension	50, 100
Size of each cluster	uniformly in [50, 500] for 4 and 10 cluster instances, and [10, 100] for 20 and 40 cluster instances

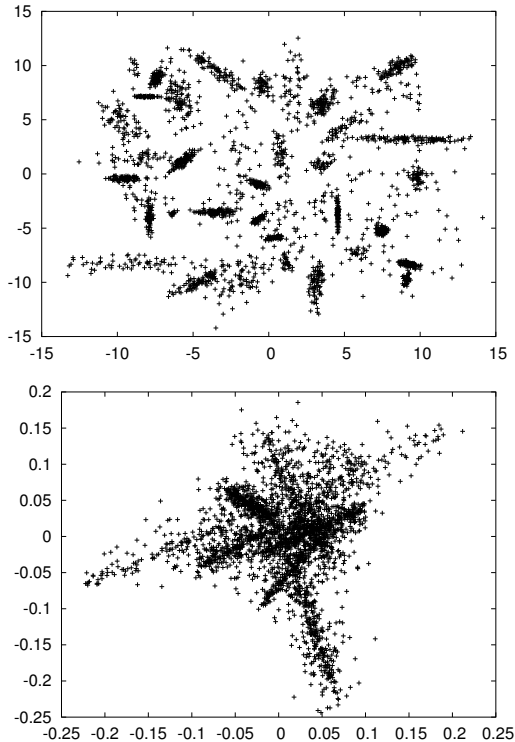


Figure 5: Examples of the data sets used in our experiments. Top: Two dimensional data set containing 40 clusters. Bottom: 100-dimensional data set containing 10 clusters (projection to two dimensions).

For each cluster, data points are generated at a Gaussian-distributed distance from a uniformly random point on the major axis, in a uniformly random direction, and are rejected if they lie outside the boundary.

After the data points of all clusters are generated (with the origin initially at $0, \dots, 0$), a genetic algorithm is used to translate the location of the origin of each cluster so that a cost consisting of the overall deviation of the entire data set, plus a penalty term for any overlapping clusters, is minimized. This has the effect of ‘arranging’ the clusters in a compact configuration (see Figure 5).

Using this method, 80 different data sets were generated, as described in Table 2.

4 Experimental setup

4.1 Alternative methods

We evaluate the new version of MOCK by comparing it to the previous version of MOCK, as well as three established single-objective clustering methods, whose implementations are described below. In order to permit a suitable comparison against MOCK, the single-objective algorithms are run for a range of different numbers of clusters $k \in [1, 50]$. We then compare both the quality of the best solution generated and the quality of solutions selected by the Silhouette Width, and by MOCK’s attainment scores, respectively.

4.1.1 k -means

Starting from a random partitioning, the k -means algorithm repeatedly (i) computes the current cluster centres (that is, the average vector of each cluster in data space) and (ii) reassigns each data item to the cluster whose centre is closest to it. It terminates when no more reassignments take place. By this means, the intra-cluster variance, that is, the sum of squares of the differences between data items and their associated cluster centres, is locally minimized.

Our implementation of the k -means algorithm is based on the batch version of k -means, that is, cluster centres are only recomputed after the reassignment of all data items. To reduce suboptimal solutions k -means is run repeatedly (10 times) using random initialisation (which is known to be an effective initialization method [10]) and only the best result in terms of intra-cluster variance is returned.

4.1.2 Hierarchical clustering

In general, agglomerative clustering algorithms start with the finest partitioning possible (that is, singletons) and, in each iteration, merge the two least distant clusters. They terminate when the target number of clusters has been obtained. Alternatively, the entire dendrogram can be generated and be cut at a later point.

Single link and average link agglomerative clustering only differ in the linkage metric used. For the linkage metric of average link, the distance between two clusters C_i and C_j is computed as the average dissimilarity between all possible pairs of data elements i and j with $i \in C_i$ and $j \in C_j$. For the linkage metric of single link, the distance between two clusters C_i and C_j is computed as the smallest dissimilarity between all possible pairs of data elements i and j with $i \in C_i$ and $j \in C_j$.

4.1.3 Silhouette Width

The Silhouette Width is a validation technique commonly used for model selection in a cluster analysis. Given clustering solutions for a range of different numbers of clusters, the Silhouette Width can be employed to determine the most appropriate solutions.

The Silhouette Width [11] for a partitioning is computed as the average Silhouette value over all data items. The Silhouette value for an individual data item i , which reflects

the confidence in this particular cluster assignment, is computed as

$$S(i) = \frac{b_i - a_i}{\max(b_i, a_i)},$$

where a_i denotes the average distance between i and all data items in the same cluster, and b_i denotes the average distance between i and all data items in the closest other cluster (which is defined as the one yielding the minimal b_i).

The Silhouette Width return values in the interval $[-1, 1]$ and is to be maximized.

4.2 Parameter settings for MOCK

Parameter settings for the new version of MOCK are given in Table 3 and are kept constant over all experiments. 100 initial solutions and 500 generations are largely sufficient for the data sets tackled in this paper, where $N \leq 5000$ and $k \leq 40$. This means that the number of total evaluations (representing the main cost of the algorithm) is 5100 (per attainment front), which is a significant reduction with regard to the previous algorithm, where a number of $\max(50, \frac{N}{20}) \times 200$ evaluations was previously used. Nevertheless, readers should be aware that, for data sets of much larger size, more evaluations may be needed. Similarly, quick convergence on data sets with very large numbers of clusters (e.g. $k \geq 50$) may require the use of a larger set of initial solutions.

In order to allow direct comparison, identical parameter settings are used for the old version of MOCK.

Table 3: Parameter settings for MOCK, where N is data set size.

<i>Parameter</i>	<i>setting</i>
Number of generations	500
External population size	1000
Internal population size	10
#(Initial solutions) <i>fsize</i>	100
Initialization	Minimum spanning tree and k -means ($L = 10$)
Mutation type	L nearest neighbours ($L = 10$)
Mutation rate p_m	$p_m = \frac{1}{N} + (\frac{1}{N})^2$
Recombination	Uniform crossover
Recombination rate p_c	0.7
Objective functions	Overall deviation and connectivity ($L = 10$)
#(Reference distributions)	1

4.3 Evaluation

Clustering quality is evaluated using the Adjusted Rand Index, an external measure of clustering quality, which is a generalization of the Rand Index.

The Rand Indices are based on counting the number of pair-wise co-assignments of data items. The Adjusted Rand Index additionally introduces a statistically induced normalization in order to yield values close to 0 for random partitions. Using a representation based on the contingency table defined by two partitionings U and V (one being the known

correct classification and one being the partition under evaluation), the Adjusted Rand Index [7] is given as

$$R(U, V) = \frac{\sum_{lk} \binom{n_{lk}}{2} - [\sum_l \binom{n_{l.}}{2}] \cdot \sum_k \binom{n_{.k}}{2}}{\frac{1}{2} [\sum_l \binom{n_{l.}}{2} + \sum_k \binom{n_{.k}}{2}] - [\sum_l \binom{n_{l.}}{2}] \cdot \sum_k \binom{n_{.k}}{2}} / \binom{n}{2},$$

where n_{lk} denotes the number of data items that have been assigned to both cluster l and cluster k , where $l \in U$ and $k \in V$. It takes values in the interval $[0, 1]$ and is to be maximized.

5 Experimental results

Table 4 and Table 5 show the results obtained by the individual algorithms. Runtimes are given in Table 6.

First, we are interested in the algorithm's capability to *generate* high-quality solutions. Table 4 gives the average Adjusted Rand Index of the best solution obtained. The results underline that the data sets used in this study are quite difficult to cluster. This difficulty is caused by two properties: the clusters are not clearly separated and are frequently of elongated shape. Evidently, this poses a problem to the three single-objective algorithms. Single link, which doesn't make any assumptions on cluster shapes but relies on good separation between clusters, generally shows a very poor performance on these data. The performance of average link and k -means, which both favour spherically shaped clusters, significantly deteriorates for those data sets with more elongated clusters (predominantly those generated by the ellipsoidal cluster generator). In contrast to this poor performance of the single-objective algorithms, the new version of MOCK generates consistently good results across the entire range of data sets.

Next, we study the performance of the algorithms, if an automated method for the *selection* of good clustering solutions is employed. For MOCK, the attainment scores of all solutions are computed and are plotted as a function of the number of clusters. For k -means, average link and single link, the Silhouette width is used instead. Again, the scores of all solutions are computed and plotted as a function of the number of clusters. Evidently, both types of plots may contain several local maxima, in particular on data sets with large numbers of clusters, where cluster structures on different levels exist. In a practical data-mining scenario all of these local maxima would be considered as 'interesting' and may be subjected to further investigation. For our evaluation purposes, we therefore select this reduced set of local maxima. The number of local maxima, and the Adjusted Rand Index of the best of the selected solutions is given in Table 4.

The results show a satisfactory performance of MOCK's method of solution selection. It reliably manages to maintain high quality solutions, while reducing the set of considered solutions to a very manageable number (often less than 10 and always less than 15, here). The Silhouette Width also shows a remarkable performance at maintaining the quality of k -means' and average-link's solutions. It should be noted, however, that the use of the Silhouette

Table 4: Number of clusters k and Adjusted Rand Index of the best solution generated by the two versions of MOCK, k -means, average link and single link. Values for MOCK and k -means are averages over 21×10 runs; average link and single link are deterministic, so averages are over the ten different problem instances only. The statistically best performer is highlighted in bold face. To test if difference were statistically significant, a paired Wilcoxon signed-rank test [1], which takes into account the dependence on problem instance, was applied to all pairs of algorithms. Notice the null hypothesis of no difference could not be rejected for MOCK and average link 10d-20c and 10d-20c at $\alpha = 0.05$.

Problem	MOCK		MOCK (old)		k -means		Average link		Single link	
	k	Rand Index	k	Rand Index	k	Rand Index	k	Rand Index	k	Rand Index
2d-4c	4.06667	0.987992	4.10952	0.98475	3.9619	0.914055	5	0.93295	18.2	0.839558
10d-4c	4.12381	0.995501	4.09524	0.99393	3.98571	0.970301	7.7	0.979746	34	0.396489
50d-4c	4	1	4.04286	0.988672	4.88095	0.561508	15.6	0.668738	26.5	0.13485
100d-4c	4	1	4.02381	0.983172	6.27619	0.491183	11.8	0.60045	11.1	0.0124546
2d-10c	10.7143	0.933782	10.2714	0.911884	11.8238	0.830955	22	0.8329	36.9	0.634749
10d-10c	13.5952	0.961277	10.3667	0.909973	9.24286	0.924482	29.3	0.938662	45.2	0.00266785
50d-10c	10.0238	0.997742	10.1714	0.9171	17.7238	0.513475	36	0.51651	29.4	0.00127714
100d-10c	10.0286	0.99976	10.2714	0.921043	16.7286	0.513788	30.5	0.582144	23.3	0.00128917
2d-20c	20.081	0.938703	19.3048	0.918921	34.6619	0.862799	26.1	0.936857	40	0.822736
10d-20c	20.4905	0.996973	20.1857	0.991951	21.4905	0.982288	20.2	0.997444	40.9	0.946316
50d-20c	21.8952	0.89365	20.881	0.616976	35.9619	0.558086	43.4	0.487253	41.6	0.00655653
100d-20c	20.9476	0.883184	20.5286	0.638534	34.6286	0.567416	46.1	0.558948	38	0.0167002
2d-40c	42.4524	0.859737	26.4143	0.671327	41.8381	0.692796	44.8	0.823132	47	0.532413
10d-40c	43.6333	0.987919	34.9476	0.872236	42.9905	0.967751	42.3	0.995724	44.8	0.745406
50d-40c	46.8619	0.808289	27.1714	0.201575	47.4524	0.463091	48.9	0.200675	41.6	0.00156619
100d-40c	47.5762	0.791701	27.1762	0.208609	47.5762	0.472898	48.3	0.228758	48.1	0.00279215

Table 5: Number of clusters k and Adjusted Rand Index of the best local maxima, and number of local maxima $\#lmax$ in a plot of the attainment scores (MOCK only) or the Silhouette Width (k -means, average link and single link). Values are averages as explained in Table 4. The statistically best performer (determined in the same way as for Table 4) is highlighted in bold face. Here, the best performer is always significantly better at beyond $\alpha = 0.001$.

Problem	MOCK			MOCK (old)			k -means			Average link			Single link		
	k	Rand Index	$\#lmax$	k	Rand Index	$\#lmax$	k	Rand Index	$\#lmax$	k	Rand Index	$\#lmax$	k	Rand Index	$\#lmax$
2d-4c	4.06667	0.959832	10.4	3.89524	0.922555	3.1381	3.88571	0.898591	1.93333	5.2	0.928867	12.7	19.1	0.839177	16.6
10d-4c	4.17143	0.993823	10.3571	4.05714	0.992991	3.32857	2.8	0.775792	1.91429	8.2	0.977117	17.4	34.8	0.396114	16.1
50d-4c	4.02857	0.978285	5.59524	5.74762	0.782369	2.50952	3.4	0.433664	1.93333	19.7	0.64199	9.64	28.1	0.130981	9.2
100d-4c	4.53333	0.93326	5.66667	5.86667	0.746277	2.50952	3.3	0.367441	1.8	16.5	0.578219	7.7	12.2	0.0120891	11.2
2d-10c	10.7714	0.913508	11.1429	9.89524	0.835167	3.9	10.181	0.783076	4.34286	25.3	0.812206	13	37.8	0.634576	13.6
10d-10c	14.3381	0.954993	14.1143	9.24286	0.825786	2.68095	8.68571	0.900679	4.76667	26.8	0.936017	16.8	43.7	0.00265018	22.2
50d-10c	11.3095	0.948293	6.98571	10	0.871272	3.08571	10.4857	0.401777	4.04762	36.9	0.51404	4.10	30.3	0.00124138	11.6
100d-10c	12.7095	0.903848	5.89048	10.0429	0.885856	3.09048	9.97143	0.413388	4.25714	32.1	0.57415	10.3	24	0.00126184	11.2
2d-20c	20.581	0.913725	7.99048	13.3	0.693207	3.50476	24.7524	0.825355	7.65714	26.5	0.936314	15.4	40.5	0.821785	12.9
10d-20c	21.5667	0.98437	7.4381	20.4905	0.962628	3.15714	19.4952	0.959979	9.45238	20.2	0.997444	16	41.5	0.945852	13.2
50d-20c	22.0286	0.873543	8.78095	19.0905	0.533363	4.85714	26.9905	0.452967	9.99524	44.9	0.483533	6.6	45.4	0.00653575	12.3
100d-20c	21.6857	0.857716	8.7	18.8952	0.565518	4.88095	25.881	0.464521	9.6619	46.4	0.55545	6.5	38	0.01669	10.6
2d-40c	42.6476	0.847592	11.1952	14.3333	0.448179	4.29524	32.8	0.652202	7.62857	45.7	0.82095	6.8	47.1	0.532114	4.5
10d-40c	44.2333	0.979675	7.74286	30.9571	0.784865	5.24286	39.3048	0.949234	16.8524	42.6	0.99552	6.4	45.2	0.745359	6.3
50d-40c	48.181	0.790105	11.5905	24.3524	0.161799	6.6381	44.919	0.378575	14.8857	49.6	0.199993	4	45.1	0.00155807	14.5
100d-40c	47.8952	0.770879	11.3905	24.1048	0.167739	6.73333	44.5333	0.382698	15.0619	48.8	0.225682	4.8	48.4	0.00277296	14.4

Table 6: Size of the problem N , and the corresponding runtimes (in seconds) for the two versions of MOCK, k -means, average link and single link (averages over all instances/runs). The time complexity of both MOCK and k -means increases linearly with the size of the data set and the dimensionality. The runtime of agglomerative clustering is independent of dimensionality (due to the use of a pre-computed dissimilarity matrix), but quadratic in the size of the data set. All timings are for obtaining the set of solutions $k \in [1, 50]$. (For agglomerative clustering and MOCK these were determined in just one run, whereas k -means had to be re-run for each k .)

Problem	N	MOCK	MOCK (old)	k -means	Average link	Single link
2d-4c	1253.5	53.4	31.2476	63.8	14	13.2
10d-4c	1108.3	67.4	46.9524	55.5	9.5	9.2
50d-4c	1045.2	112.2	68.4571	74.7	9.6	9.2
100d-4c	1117.5	191.6	104.929	111.8	10.8	10.5
2d-10c	3120.2	162.5	91.4524	425.5	144.2	144.4
10d-10c	2955.9	241.2	134.124	416.6	136.6	136.1
50d-10c	2796.4	402.5	194.171	440.1	120.4	120.2
100d-10c	2773.9	668.6	299.257	530.1	116.9	114.5
2d-20c	1235.7	50.1	28.4667	55.4	11.5	11.7
10d-20c	1171.9	71	44.881	59.2	10.5	10.7
50d-20c	1171.1	132.1	75.6048	85.7	10.7	10.1
100d-20c	1124.9	208.3	108.043	109.9	9.8	9.2
2d-40c	2268.1	104.4	60.6	203.9	60.9	59.5
10d-40c	2355.5	154.4	92.481	250.3	73.6	72.8
50d-40c	2159.7	273.9	151.29	264	56.9	56.2
100d-40c	2106.2	451.8	236.133	316	53.2	52.8

Width as a selection criterion for MOCK is inappropriate: like k -means and average link, the Silhouette Width is biased towards compact, spherically-shaped clusters and leads to poor results when applied to MOCK's solutions (results not shown).

In comparison with the previous version of MOCK, runtimes are approximately doubled. However, the performance of the old version remains poor when given equivalent runtimes.

6 Conclusion

Our principal goal in this work has been the extension of multiobjective clustering to the application to large and complex data sets. In order to achieve this goal, three modifications to our algorithm MOCK have been introduced, which significantly improve the convergence speed of the algorithm and permit a substantial reduction in the number of total evaluations required.

We also conducted the first evaluation of multiobjective clustering on large high-dimensional data sets with up to 5000 data items, 100 dimensions and 40 clusters. Due to the lack of publicly available and sufficiently complex clustering test suites, two new data generators have been developed for this purpose. The results obtained suggest that the advantages of multiobjective clustering should scale up to complex data-mining scenarios.

MOCK currently features a simple visual interface permitting the interactive exploration of the Pareto front and the solutions discovered. The development of more powerful and scalable visualization approaches is under way. The software for MOCK is available at <http://dbk.ch.umist.ac.uk/handl/mock/>.

Acknowledgements: JH gratefully acknowledges support of a scholarship from the Gottlieb Daimler- and Karl Benz-Foundation, Germany. JK is supported by a David Phillips Fellowship from the Biotechnology and Biological Sciences Research Council (BBSRC), UK.

Bibliography

- [1] W. J. Conover. *Practical Nonparametric Statistics, second edition*. John Wiley & Sons, New York, 1980.
- [2] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 283–290, San Francisco, California, 2001. Morgan Kaufmann Publishers.
- [3] V. Estivill-Castro. Why so many clustering algorithms: A position paper. *ACM SIGKDD Explorations Newsletter Archive*, 4:65–75, 2002.
- [4] J. Handl and J. Knowles. Evolutionary multiobjective clustering. In *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pages 1081–1091. Springer, Heidelberg, 2004.
- [5] J. Handl and J. Knowles. Multiobjective clustering with automatic determination of the number of clusters. Technical Report TR-COMPSYSBIO-2004-02, UMIST, Manchester, UK, 2004.
- [6] J. Handl and J. Knowles. Exploiting the trade-off: the benefits of multiple objectives in data clustering. In *Proceedings of the Third International Conference on Evolutionary Multicriterion Optimization*, pages 547–560. Springer-Verlag, 2005.
- [7] A. Hubert. Comparing partitions. *Journal of Classification*, 2:193–198, 1985.
- [8] J. Kleinberg. An impossibility theorem for clustering. In *Proceedings of the 15th Conference on Neural Information Processing Systems*, Vancouver, Canada, 2002.
- [9] Y.-J. Park and M.-S. Song. A genetic algorithm for clustering problems. In *Proceedings of the Third Annual Conference on Genetic Programming*, pages 568–575, Madison, WI, 1998. Morgan Kaufmann.
- [10] J. M. Pena, J. A. Lozana, and P. Larranaga. An empirical comparison of four initialization methods for the k -means algorithm. *Pattern Recognition Letters*, 20:1027–1040, 1999.
- [11] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
- [12] W. S. Sarle. Cubic clustering criterion. Technical report, SAS Technical Report A-108, Cary, NC: SAS Institute Inc, 1983.
- [13] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research*, 3:583–617, 2002.
- [14] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the Gap statistic. Technical report, Stanford University, 2000.