

An Evolutionary Approach to Multiobjective Clustering

Julia Handl and Joshua Knowles

Abstract—The framework of multiobjective optimization is used to tackle the unsupervised learning problem, data clustering, following a formulation first proposed in the statistics literature. The conceptual advantages of the multiobjective formulation are discussed and an evolutionary approach to the problem is developed. The resulting algorithm, multiobjective clustering with automatic k -determination, is compared with a number of well-established single-objective clustering algorithms, a modern ensemble technique, and two methods of model selection. The experiments demonstrate that the conceptual advantages of multiobjective clustering translate into practical and scalable performance benefits.

Index Terms—Clustering, determination of the number of clusters, evolutionary clustering, model selection, multiobjective clustering.

I. INTRODUCTION

MOST respected texts on data clustering (e.g., [1]–[3]) define the problem informally, e.g., Arabie *et al.* [4] define clustering as:

“Those methods concerned in some way with the identification of homogeneous groups of objects,”

while [2] documents several different definitions including this one of a cluster:

“A cluster is a set of entities that are alike, and entities from different clusters are not alike.”

The use of informal definitions such as these reflects one of the prevailing and fundamental problems in cluster analysis: the difficulty of providing a single formal (but sufficiently broad) definition of clustering and of the concept of a cluster. In fact, Everitt points out that “such formal definition is not only difficult but may even be misplaced” [2, p. 6]. This is because the concept of a cluster is a generalization (to arbitrary dimensions) of what humans perceive, in two or three dimensions, as densely connected “patches” or “clouds” within data space, a human intuition

which is inherently difficult to capture by means of individual objective criteria. We illustrate this in Fig. 1.

While clustering algorithms are designed to capture the human notion of a cluster, their ultimate purpose is their application to the unsupervised classification of complex data sets *not interpretable* by humans. However, assessing the performance of a clustering algorithm is a problem on its own: due the lack of a formal definition of clustering, no objective performance criterion exists, unless human expert knowledge is taken into account. For this reason, [5] argues that the assessment of clustering algorithms should be based on simulation and Monte Carlo studies that “allow the researcher to know the exact cluster structure underlying the data.” In such a scenario “no doubt exists as to the true clustering, or the extent to which any given clustering algorithm has recovered this structure.” In this paper, we adopt Milligan’s strategy [5] to define the quality of a clustering, and to assess the performance of individual algorithms.

A. Single-Objective Clustering

We have outlined above that the quality of a clustering is ultimately defined in terms of external expert knowledge. However, clustering algorithms are unsupervised methods and external knowledge of any kind is not usually available during the clustering process. In order to assess the quality of individual partitionings, clustering algorithms therefore rely on the adoption of internal criteria as *proxies* for the unknown “correct classification.” The resulting transformed clustering problem (Ω, P) can then be formally defined as an optimization problem [6]: determine the clustering C^* for which

$$P(C^*) = \min_{C \in \Omega} P(C) \quad (1)$$

where Ω is the set of feasible clusterings,¹ C is a clustering of a given set of data E and $P : \Omega \rightarrow \mathbb{R}$ is the (internal) criterion function. P is always based on the notion of similarity or dissimilarity between data items, which is either provided as the input or can be derived (through a suitable distance function) if the data are embedded in a metric space.² We assume P is to be minimized without loss of generality.

Different formulations of the clustering problem vary in the optimization criterion P used, as discussed further in Section II-A. Importantly, though, most existing clustering methods attempt, explicitly or otherwise, to optimize just *one*

¹Here, we take this to be any proper partitioning of E ; we do not consider fuzzy partitions.

²In this paper, we assume data are initially embedded in a metric space, however, our method can be adapted to the case of similarity data, as shown in [7].

Manuscript received July 4, 2005; revised September 14, 2005, November 28, 2005, and January 31, 2006. The work of J. Handl was supported in part by the German Academic Exchange Service (DAAD) and in part by the Gottlieb Daimler-Foundation and Karl Benz-Foundation, Germany, under a Doctoral Scholarship. The work of J. Knowles was supported in part by the Biotechnology and Biological Sciences Research Council (BBSRC), U.K., under a D. Phillips Fellowship.

The authors are with the Manchester Interdisciplinary Biocentre, University of Manchester, Manchester M1 7ND, U.K. (e-mail: j.handl@postgrad.manchester.ac.uk; j.knowles@manchester.ac.uk).

Digital Object Identifier 10.1109/TEVC.2006.877146

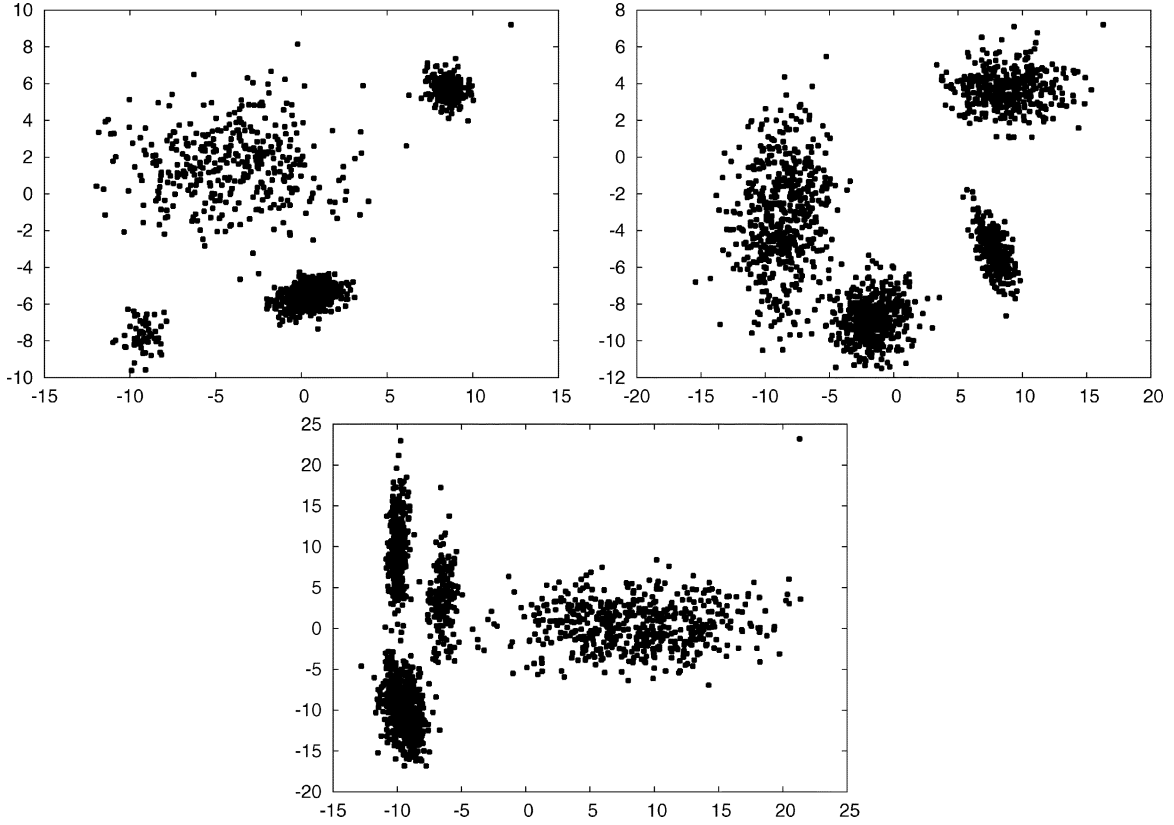


Fig. 1. Different cluster structures in two dimensions. Clusters may be *inter alia* unequally sized, differently shaped, or slightly overlapping. While each of these features may pose problems to specific clustering criteria, the main cluster structure can be perceived easily by humans. Here, each figure contains precisely four clusters, each generated by a normal distribution. Notably, normal distributions are just one possible and simplified cluster model, and clusters can take more complex (e.g., nonconvex) shapes.

such criterion, and it is this confinement to a particular cluster property that explains the fundamental discrepancies observable between the solutions produced by different algorithms on the same data, and will cause a clustering method to fail (as judged by means of external knowledge) in a context where the criterion employed is inappropriate.

B. Multiobjective Clustering

In practice, the problem of choosing an appropriate clustering objective (viz. algorithm) can be alleviated through the application and comparison of multiple clustering methods [8], or through the *a posteriori* combination of different clustering results by means of ensemble methods [9], [10]. However, a more principled approach may be the consideration of clustering as a multiobjective optimization problem, as suggested by [6]. In a multiobjective clustering problem $(\Omega, P_1, P_2, \dots, P_m)$ [6], we aim to determine the clustering C^* for which

$$P_t(C^*) = \min_{C \in \Omega} P_t(C), \quad t = 1, \dots, m \quad (2)$$

where Ω is the set of feasible clusterings, C is a clustering of a given set of data E , and P_t , $t = 1, \dots, m$ is a set of m different (single) criterion functions. Usually, no single best solution for this optimization task exists, but instead, the framework of Pareto optimality is embraced. For two solutions C_1 ,

$C_2 \in \Omega$, solution C_1 is said to *dominate* solution C_2 (denoted as $C_1 \prec C_2$) if and only if

$$\forall t : P_t(C_1) \leq P_t(C_2) \wedge \exists t : P_t(C_1) < P_t(C_2). \quad (3)$$

The set of Pareto-optimal solutions or Pareto-efficient solutions Π is then defined as

$$\Pi = \{C \in \Omega : \nexists C' \in \Omega : C' \prec C\} \quad (4)$$

and the image of this set in the objective space is referred to as the Pareto front.

The set of Pareto-optimal solutions to a multiobjective clustering problem $(\Omega, P_1, P_2, \dots, P_m)$ always comprises the optimal solutions to the single-objective clustering problems $(\Omega, P_1), (\Omega, P_2), \dots, (\Omega, P_m)$. For *ideal* single- and multi-objective clustering algorithms (i.e., algorithms that always identify all globally optimal solutions, and the entire Pareto-optimal set, respectively), we therefore trivially know that the multiobjective algorithm will *always* find a solution *as good or better* (equal in terms of the clustering objective optimized and equal or *possibly* better in terms of external knowledge) than those of the single-objective algorithms. In situations where the best solution corresponds to a tradeoff between the different objectives *only the multiobjective clustering algorithm* will

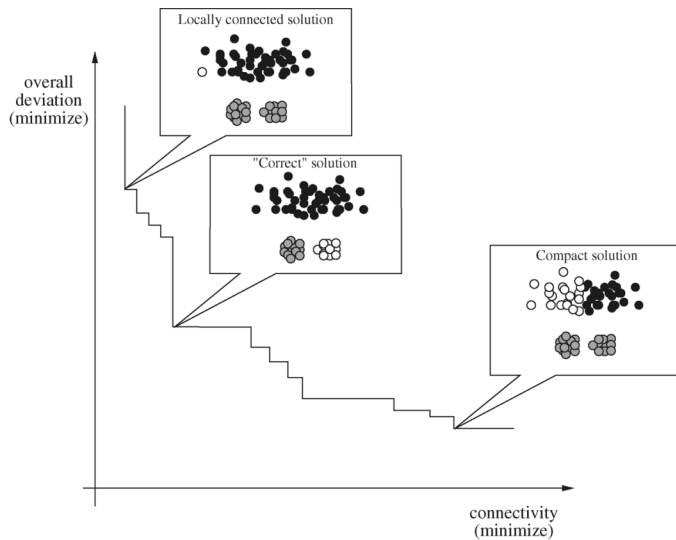


Fig. 2. The correct clustering solution often corresponds to a tradeoff between two or more clustering objectives. A Pareto front (depicted as a line) and three different Pareto optimal clustering solutions are shown for a simple three-cluster data set, plotted in two-objective space. The solution to the top left is generated by an algorithm like single link agglomerative clustering, which minimizes *connectivity* only. The solution to the bottom right is generated by an algorithm like *k*-means, which minimizes *overall deviation* only. (Connectivity and overall deviation are defined in Section III-A3. Also, see Section II.) The correct solution is situated some way between these two solutions and so it would not usually be discovered by either method. A multiobjective approach considering the tradeoffs between the two objectives should be able to access this solution much more readily. For sake of clarity, the approximation set, in this example, only contains solutions for $k = 3$. More generally, the number of clusters can also be kept dynamic—in this case, an approximation set is obtained in which the number of clusters varies along the Pareto front.

be able to find it.³ An example of a data set, for which this is relevant is shown in Fig. 2. Here, different possible clustering solutions are plotted in two-objective space and it can be seen that the correct solution represents a tradeoff between the two objectives.

C. Scope of This Paper

The theoretical advantages of multiobjective clustering outlined above suggest a sound underlying basis for a novel clustering method. However, a previous implementation of the idea given in [6] was limited to a mere proof-of-concept, and no other realizations have been proposed until our recent work [7], [11]–[14]. This work is summarized in Section II-D.

In this paper, we provide the first detailed and self-contained description of the resulting and currently most advanced version of multiobjective clustering, multiobjective clustering with automatic *k*-determination (MOCK) (all our previous papers on MOCK [7], [13], [14] refer to an unrefereed technical report [12] for algorithm details). Also, for the first time, we discuss the conceptual and practical advantages of multiobjective clustering in detail, and relate our work to that by [6]. In addition, we provide a comprehensive evaluation and comparison of the algorithm. For this purpose, we separate the tasks of

³In the case of iterative search algorithms, it is *possible* that a single-objective approach would visit such tradeoff solutions during a run, but it would not recognize them as good and would almost surely discard them.

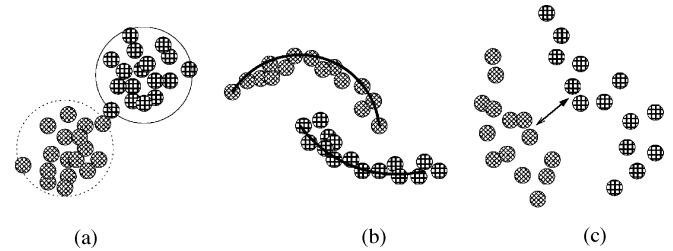


Fig. 3. Illustration of the three different categories of clustering objectives. Clearly, objectives based on compactness and connectedness make very different assumptions about the properties of a data set and usually result in very different clustering results. In contrast, connectedness and spatial separation are somewhat related (opposed) concepts. (a) Compactness. (b) Connectedness. (c) Spatial separation.

clustering and of model selection and evaluate MOCK's performance on both of these tasks individually. In particular, we compare the algorithm against three single-objective clustering methods, one advanced cluster ensemble method, and two established approaches to model selection. The results, all new to this paper, demonstrate that a multiobjective approach can have distinct benefits both in the context of clustering and model selection.

The remainder of this paper is organized as follows. Section II discusses related work and our own previous work on multiobjective clustering. Section III describes our multiobjective clustering algorithm, MOCK, in detail. This is followed by two experimental sections: Section IV describes the experimental setup, and Section V summarizes results. Section VI discusses the main findings of this work and Section VII concludes.

II. RELATED WORK

A. Clustering Algorithms Categorized by Criterion Optimized

Traditional classifications of clustering algorithms primarily distinguish between *hierarchical*, *partitioning*, and *density-based* methods [1], [2], [15]. Here, we will use a somewhat different categorization that is based on the clustering criterion (implicitly or explicitly) optimized by the algorithm [8]. With regard to this, existing clustering algorithms fall into three major groups, illustrated in Fig. 3. First, algorithms striving for *compact* clusters, a concept which is generally implemented by keeping intracluster variation (i.e., variation between same-cluster data items or between data items and cluster representatives) small. This category includes algorithms like *k*-means [16], average link agglomerative clustering [17], self-organizing maps [18], or model-based clustering approaches [19], [20]. The resulting methods tend to be very effective for spherical and/or well-separated clusters, but they may fail for more complicated cluster structures.

Methods based on a concept of *connectedness* make up the second group. They employ a more local concept of clustering based on the idea that neighboring data items should share the same cluster. Algorithms implementing this principle are density-based methods [21], [22] and methods like single link agglomerative clustering [17]. These are well-suited to detect clusters of arbitrary shapes, however, they can lack robustness when there is little spatial separation between clusters.

Objectives based on *spatial separation* can be identified as the underlying criteria for a further class of clustering algorithm. However, spatial separation on its own is a criterion that gives little guidance during the clustering process and can easily lead to trivial solutions (e.g., classifying outliers as individual clusters, but merging the bulk of data items into one cluster). It is therefore usually combined with other objectives, most notably measures of compactness, or “balancedness” of cluster sizes. Unlike the first two groups, the resulting clustering objectives (e.g., Dunn Index, Davies–Bouldin Index [23]) have not been used as the basis for any specialized clustering heuristic, but can be used explicitly as the objective in a general-purpose metaheuristic.

B. Evolutionary Algorithms for Single-Objective Clustering

Evolutionary algorithms (EAs) have been the metaheuristic most frequently used for clustering. However, previous work by other researchers has been limited to the single-objective case: criteria based on cluster compactness have been the objectives most commonly employed, as these measures provide smooth incremental guidance in all parts of the search space.

A variety of different EA representations for clustering solutions have been explored in the literature, ranging from a straightforward encoding (with the i th gene coding for the cluster membership of the i th data item), to more complex representations, such as is used in Falkenauer’s grouping genetic algorithm (GA) [24]. However, none of these direct encodings significantly reduces the size of the clustering search space, making it crucial to design effective and efficient search operators. For this reason, many researchers have chosen to use a more indirect approach that borrows from the popular k -means algorithm: the representation codes for cluster centroids/medoids only; each data item is subsequently assigned to the closest cluster representative [25]–[28].

Genuine hybridizations between EAs and k -means have also been introduced, and are of particular interest with respect to feature selection for unsupervised classification. In a number of papers [29]–[31], EAs have been used to evolve the features serving as the input for the k -means algorithm. The k -means’ clustering solutions are then evaluated and the resulting objective values (under one or more suitable clustering criteria) are fed back to the EA. In this context, the use of multiple criteria is desirable in order to identify possible tradeoffs between feature set size and solution quality, and multiobjective evolutionary algorithms (MOEAs) have therefore been employed.

C. Ensemble Techniques

In contrast to their application for feature selection, MOEAs have not been applied for the actual clustering task by other researchers. This is despite the general agreement that clustering objectives can be conflicting or complementary, and that no single clustering objective can deal with every conceivable cluster structure [32], [33]. To date, most attempts to deal with this problem have focused on the *retrospective* combination of different clustering results by means of ensemble methods [9], [10], [34]–[37]. In order to construct clustering ensembles, different clustering results are retrieved by repeatedly running the

same algorithm (using different initializations, parameterizations, bootstrapping, or a varying number of clusters) or several complementary methods (e.g., agglomerative algorithms based on diverse linkage criteria such as single link and average link). The resulting solutions are then combined into an ensemble clustering using graph-based approaches [9], expectation maximization [36] or co-association methods [36].

Results reported in the literature demonstrate that clustering ensembles are often more robust and yield higher quality results than individual clustering methods, indicating that the combination of several clustering objectives is favorable. However, ensemble methods do not fully exploit the potential of using several objectives: as they are limited to the *a posteriori* integration of the solutions returned by the individual algorithms, they cannot explore tradeoff solutions *during* the clustering process.

Tackling clustering as a truly multiobjective optimization problem may be a more flexible approach (see Section I-B). The potential of multiobjective data clustering has been previously pointed out in [6] and [38] both with respect to a set of different clustering objectives [6] and with respect to a set of different dissimilarity matrices [6], [38]. However, until our own recent work, practical implementations of multiobjective clustering have been limited to a proof-of-concept on data sets of very limited size (e.g., 12 data items).

D. Evolutionary Algorithms for Multiobjective Clustering

The first multiobjective clustering algorithm that we developed, Voronoi initialized evolutionary nearest-neighbor algorithm (VIENNA [11]), is based on PESA-II [39], and uses two objectives. It employs a straightforward encoding of a clustering, with a gene for each data item and its allele value specifying the cluster to which the data item should belong. An advanced initialization scheme based on Voronoi cells and directed mutation is needed to make up for deficiencies in this encoding, and crossover is found to be too disruptive to use. Moreover, VIENNA needs to be provided with the correct number of clusters and does not provide any means of selection of good solutions from the final Pareto front.

In subsequent work, we fine-tuned one of the objectives used in VIENNA, and found a better encoding that does not fix the number of clusters and, because of good locality and heritability, allows a much more effective exploration of the search space via suitable operators. We also developed a method of selecting solutions from the Pareto front based on a null model, thus also determining the number of clusters, automatically. These developments were incorporated in a new algorithm called MOCK, first described in an unreferenced technical report [12]. The report also includes an evaluation of MOCK and it is possible to see substantial improvement over the results for VIENNA [11], resulting primarily from the improved encoding.

In [13], a brief summary of MOCK is given and we use a canonical example problem to demonstrate how the best solution to some clustering problems is a tradeoff between two objectives and cannot be reached by methods optimizing these objectives individually. MOCK is further extended in [14] in order to improve its scalability to large, high-dimensional data sets, and data with large numbers of clusters: this involves improvements to both the initialization and the null model used, which

are compared empirically with the earlier versions. Further work has also resulted in the introduction of MOCK-around-medoids, which allows for the clustering of similarity data [7] (as opposed to vectorial data, i.e., points in a metric space).

III. MOCK

In this section, we describe in detail our multiobjective clustering algorithm MOCK. It consists of two main phases: In its initial *clustering phase*, MOCK uses a MOEA to optimize two complementary clustering objectives. The output of this first phase is a set of mutually nondominated clustering solutions, which correspond to different tradeoffs between the two objectives, and also to different numbers of clusters. In the second *model-selection phase*, MOCK analyzes the shape of the tradeoff curve and compares it to the tradeoffs obtained for an appropriate null model (i.e., by clustering random data). Based on this analysis, the algorithm provides an estimate of the quality of all individual clustering solutions, and determines a set of potentially promising clustering solutions. Often, a single solution is clearly preferred and, in these cases, the number of clusters inherent to the data set, k , is thus estimated implicitly.

A. Phase I: Clustering

EAs are well-suited for multiobjective optimization as their use of a population enables the whole Pareto front to be approximated in a single algorithm run. Thus, in recent years, there has been a growing effort in applying evolutionary computing methods in multiobjective optimization, giving rise to many different algorithms (see, e.g., [40]–[42] for reviews of the state of the art, and [43] for an extensive list of references on this field). One example of a MOEA, the Pareto envelope-based selection algorithm version 2 (PESA-II) [39], forms the basis of MOCK's clustering phase.

1) *PESA-II*: PESA-II follows the standard principles of an EA with the difference that *two* populations of solutions are maintained: an internal population (*IP*) of fixed size, and an external population (*EP*) of nonfixed but limited size. The purpose of the EP is to *exploit* good solutions: to this end it implements a form of elitism by maintaining a large and diverse set of the nondominated solutions discovered during search. The IP's job, on the other hand, is to *explore* new solutions, and it achieves this by the standard EA processes of reproduction and variation (i.e., cloning, recombination, and mutation). Selection occurs at the interface between the two populations, both in the update of the EP and in the construction of the IP, which is done anew each generation. The solutions in EP are stored in "niches," implemented as a hypergrid of equally sized cells in the objective space. This is sometimes referred to as a histogram approach to density estimation [44]. A tally of the number of solutions that occupy each niche is kept and this is used to encourage solutions to cover the whole objective space, rather than bunch together in one region. To this end, a nondominated solution that tries to enter a full EP can only do so if it occupies a less crowded niche than some other solution (or if it dominates a solution in the EP). Moreover, when the IP of each generation is constructed from the EP, solutions are selected uniformly from among the populated niches—thus highly populated niches do not contribute more solutions than less populated ones. An important advantage of PESA-II is that this niching policy uses an

adaptive range equalization and normalization of the objective function values. This means that difficult parameter tuning is avoided, and objective functions that have very different ranges can be readily used. PESA-II can also handle any number of objective functions, though its performance (like that of any other Pareto-based MOEA) is likely to deteriorate with larger numbers of objectives [45]. For further details on PESA-II, the reader is referred to [39].

2) *Adaptation of PESA-II for Clustering*: The application of PESA-II to the clustering problem requires the choice of:

- two or more objective functions;
- a suitable genetic encoding of a partitioning;
- one or more genetic variation operators (e.g., mutation and/or crossover);
- an effective initialization scheme.

These choices are nontrivial and are crucial for the performance and particularly for the scalability of the algorithm: many encodings for clustering work well for data sets with a few tens or hundreds of data points, but their performance breaks down rapidly for larger data sets. The design of an effective EA for clustering requires a close harmonization of the encoding, the operators, and the objective functions, so that the effective search space is reduced and the search can be guided effectively. Our choices for these components, determined after extensive experimentation (cf. [11] and [14]) are described next.

3) *Objective Functions*: For the clustering objectives, we are interested in selecting optimization criteria that reflect fundamentally different aspects of a good clustering solution. From the groups identified in Section II-A, we therefore select two complementary objectives: one based on compactness, the other one based on connectedness of clusters. We refrain from using a third objective based on spatial separation, as the concept of spatial separation is intrinsic (opposite) to that of connectedness of clusters.

In order to express cluster compactness, we compute the *overall deviation* of a partitioning. This is simply computed as the overall summed distances between data items and their corresponding cluster center

$$\text{Dev}(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k) \quad (5)$$

where C is the set of all clusters, μ_k is the centroid of cluster C_k , and $\delta(.,.)$ is the chosen distance function (here, the Euclidean distance). As an objective, overall deviation should be minimized. This criterion is similar to the popular criterion of intracluster variance, which squares the distance value $\delta(.,.)$ and is more strongly biased towards spherically shaped clusters.

As an objective reflecting cluster connectedness, we use a measure, connectivity, which evaluates the degree to which neighboring data points have been placed in the same cluster. It is computed as

$$\text{Conn}(C) = \sum_{i=1}^N \left(\sum_{j=1}^L x_{i,nn_{ij}} \right) \quad (6)$$

where $x_{r,s} = \begin{cases} \frac{1}{j}, & \text{if } \exists C_k : r \in C_k \wedge s \in C_k \\ 0, & \text{otherwise} \end{cases}$

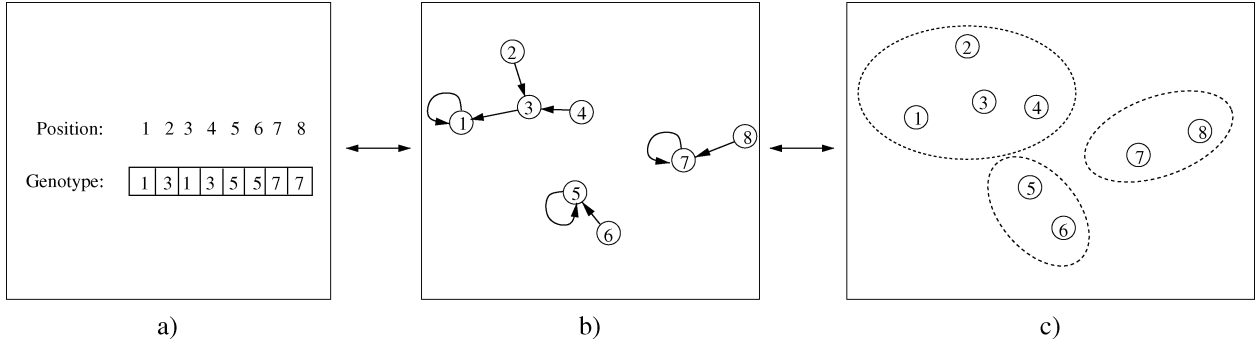


Fig. 4. Illustration of the locus-based adjacency scheme. Here, a data set consisting of eight data items (numbered from 1 to 8) is clustered. (a) Shows one (out of many) possible genotypes that is translated into the graph structure given in (b) (the graph is shown as directed only to aid in understanding how it originates from the genotype). Every connected component within this graph is finally interpreted as an individual cluster, as visualized by the ellipses in (c).

nn_{ij} is the j th nearest neighbor of datum i , N is the size of the clustered data set, and L is a parameter determining the number of neighbors that contribute to the connectivity measure. As an objective, connectivity should be minimized. This measure is conceptually similar to the criterion of nearest-neighbor consistency introduced by [46]. The main difference is the use of a gradually decreasing penalty ($1/j$), where Ding and He use a constant; our approach gives more emphasis to the nearest neighbors. It therefore permits a finer distinction between the quality of clustering solutions and allows for the identification of clusters of sizes significantly smaller than L .

Overall deviation (like our mutation operator, see Section III-A6) requires the one-off computation of the nearest-neighbor lists in the initialization phase. Subsequently, both objectives, overall deviation and connectedness, can be efficiently computed in linear time.

A further important aspect in the choice of these objective functions is their potential to balance each other's tendency to increase or decrease the number of clusters, enabling the use of a representation that does not prespecify a number of clusters k . While the objective value associated with overall deviation necessarily improves with an increasing number of clusters, the opposite is the case for connectivity. The interaction of the two is crucially important in order to keep the number of clusters dynamic, avoid convergence to trivial solutions (which would be, e.g., N singleton clusters for overall deviation and a one-cluster solution for connectivity), and explore interesting areas of the solution space (cf. [13]).

4) *Genetic Representation and Operators:* We employ the locus-based adjacency representation proposed by [47] and illustrated in Fig. 4. In this graph-based representation, each individual g consists of N genes g_1, \dots, g_N 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 connected components. All data items belonging to the same connected component are then assigned to one cluster. Note that this decoding step can be performed in linear time [48].

The locus-based adjacency encoding scheme has several major advantages for our application. Most importantly, there is no need to fix the number of clusters in advance, as it is automatically determined in the decoding step. Hence, we can

evolve and compare solutions with different numbers of clusters in just one run of the EA. Furthermore, the representation is well-suited for the use with standard crossover operators such as uniform, one-point or two-point crossover. In more traditional encodings for clustering, these straightforward crossover operators are usually highly disruptive, and therefore detrimental for the clustering process. In a link-based encoding, in contrast, they effortlessly implement merging and splitting operations on individual clusters, while maintaining the remainder of the partitioning.

Note here that we cannot use an encoding based on cluster centres even though these have been the most successful in the EA clustering literature [25]–[28]. This is because such an encoding does not allow the evolution of solutions in conflict with its underlying spherical cluster model and would therefore be strongly biased toward the objective of overall deviation. Thus, on data sets where the two objectives are substantially in conflict (such as Smile, Spiral, or Long, cf. Section IV-C), it would not permit the identification of good connectivity solutions. In order to explore the tradeoffs between the two objectives effectively, we need to use an encoding without this model bias.

5) *Uniform Crossover:* We choose the uniform crossover [49] in favor of one- or two-point because it is unbiased with respect to the ordering of genes and can generate any combination of alleles from the two parents (in a single crossover event) [50]. An example of the operation of uniform crossover on the encoding employed is shown in Fig. 5; the good heritability of the encoding under this crossover operator can be seen.

6) *Neighborhood-Biased Mutation Operator:* Despite its advantages regarding crossover, the locus-based adjacency scheme has been little used in the literature, owing to the size of the resulting search space, which, for a data set of size N , is of size N^N . However, the size of the search space “seen” by mutation can be significantly reduced using a suitable mutation operator. We use a restricted *nearest-neighbor mutation*, where each data item can only be linked to one of its L nearest neighbors. Hence, $g_i \in \{nn_{i1}, \dots, nn_{iL}\}$, where nn_{il} denotes the l th nearest neighbor of data item i . This reduces the extent of the search space seen by mutation to just L^N with $L \ll N$, typically. Note that the nearest-neighbor list is precomputed only once in the initialization phase of the algorithm.

The properties of the encoding can additionally be employed to bias the mutation probabilities of individual genes. Intuitively, “longer links” in the encoding are expected to be less

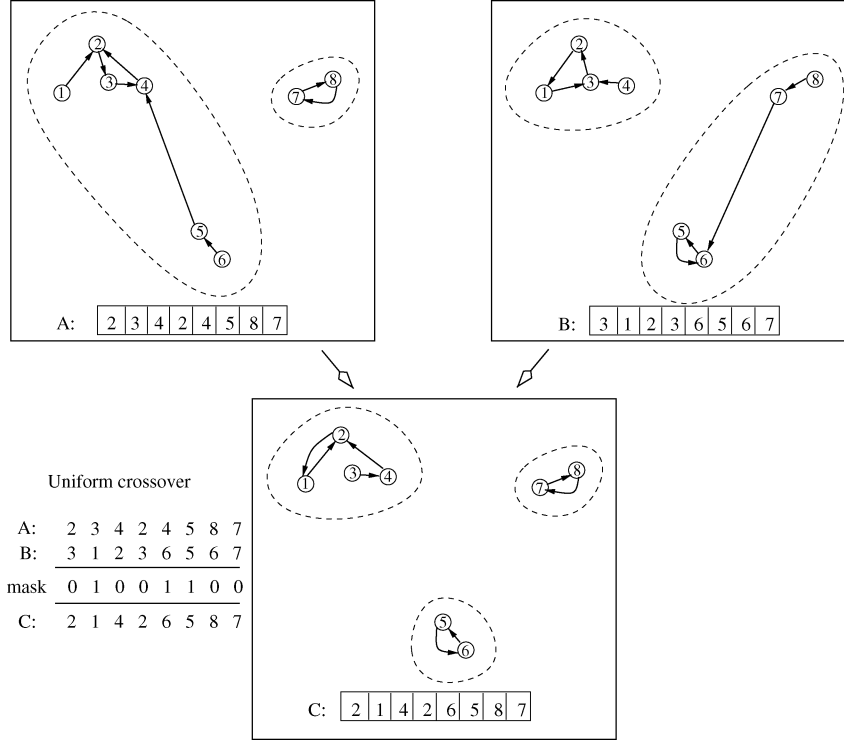


Fig. 5. Two parent partitionings, their graph structure, and their respective genotypes, *A* and *B* are shown. A standard uniform crossover of the genotypes yields the child *C*, which has inherited much of its structure from its parents, but differs from both of them.

favorable, 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

$$p_m = \frac{1}{N} + \left(\frac{l}{N}\right)^2 \quad (7)$$

where $j = nn_{il}$ and N is the size of the data set. This helps to quickly discard unfavorable links, and encourages the exploration of solutions with large numbers of clusters.

7) Initialization: Our initialization routine is motivated by the observation that different single-objective clustering algorithms tend to perform well (find good approximations) in different regions of the Pareto front (see [13] and Fig. 6). In particular, algorithms consistent with connectivity tend to generate close to optimal solutions in those regions of the Pareto front where connectivity is low, whereas algorithms based on compactness perform well in the regions where overall deviation is low. We therefore use an initialization based on two different single-objective algorithms, in order to obtain a good initial spread of solutions and a close initial approximation to the Pareto front (cf. [14] for an evaluation of the performance of this initialization scheme).

Solutions performing well under connectivity are generated using minimum spanning trees (MSTs). The use of MSTs has the advantage that the links present in a given MST can be translated directly into the locus-based adjacency encoding that the EA operates with. Solutions performing well under overall deviation are generated using the k -means algorithm. The translation of these flat (that is, nonhierarchical) partitionings into

the locus-based adjacency encoding is slightly more involved, as described next.

Generation of Interesting MST Solutions: Making use of the complete dissimilarity matrix of the given data set, we first compute the complete MST using Prim's algorithm [51]. As the complete MST corresponds to a one-cluster solution, we are then interested in obtaining a range of good solutions with different numbers of clusters. Simply removing the longest links from this MST does not yield the desired results: in the absence of spatially separated clusters, this would isolate outliers such that many of the solutions generated would be highly similar (like the single link solutions in Fig. 6).

In order to avoid this effect, 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 structure.

Definition 1: A link $i \rightarrow j$ is considered as *interesting*, iff $i = nn_{jl} \wedge j = nn_{ik} \wedge l > L \wedge k > L$, where L is the cardinality of the set of nearest neighbors (see Section III-A6). Its *degree of interestingness* is $d = \min(l, k)$. Informally, this means, that a link between two items i and j is considered interesting if neither of them is a part of the other item's set of L nearest neighbors (see Fig. 7).

Definition 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

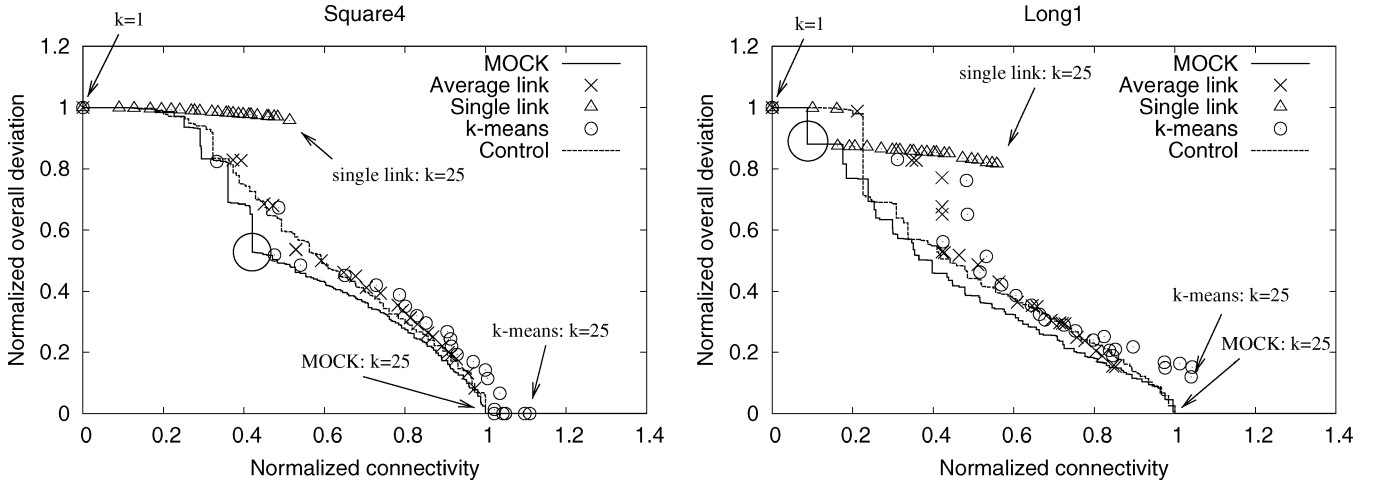


Fig. 6. Comparison of the output of MOCK with the solutions obtained by k -means, average link, and single link agglomerative clustering on the *Square4* and *Long1* data sets (cf. Section IV-C), when run for a range of different numbers of clusters $k \in 1, \dots, 25$. Solutions have been evaluated using overall deviation and connectivity, and visualized in normalized two-objective space. Both MOCK's Pareto front and one of its control fronts are visualized as attainment surfaces (i.e., points are joined up with a stepped line), and arrows indicate the position of the $k = 1$ solution (identical for all algorithms) and some of the $k = 25$ solutions (different for each algorithm). Knees are indicated by circles centered around the solution identified by MOCK. The single-objective algorithms are less successful than MOCK at finding good tradeoffs between the two objectives, but outperform MOCK in terms of the optimization of their own (single) objective, e.g., comparing the right-most single link solution (which corresponds to $k = 25$) to MOCK's $k = 25$ solution, it is clear that single link's solution has the better connectivity value.

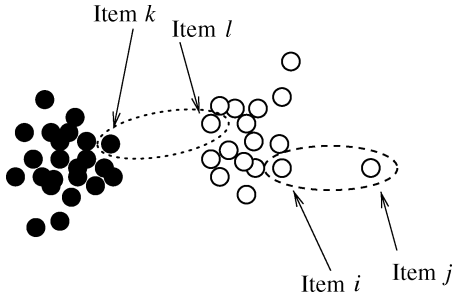


Fig. 7. Distinction between a “long” link connecting an outlier to a cluster, and a “long” link connecting two clusters. Item i is within the set of nearest neighbors of item j , and the links $i \rightarrow j$ or $j \rightarrow i$ are therefore considered uninteresting. In contrast, both data items l and k are not within each other's set of nearest neighbors, and thus the links $l \rightarrow k$ or $k \rightarrow l$ are interesting.

list, a set of clustering solutions is then constructed: for $n \in \{0, \dots, \min(I, \lfloor 0.5 \cdot fsize \rfloor - 1)\}$, where $fsize$ is the total number of initial solutions, clustering solution C_n is generated by removing the first n interesting links (i.e., at most $1/2$ of the initial population consists of MST-derived solutions). A link removed at position i is subsequently replaced by a link to a randomly chosen neighbor j with $j = nn_{il} \wedge l \leq L$.

Generation of k -Means Solutions: Finally, we consider the generation of k -means solutions. We start by running the k -means algorithm (for only ten iterations) for different numbers of clusters $k \in \{2, \dots, fsize - \min(I, \lfloor 0.5 \cdot fsize \rfloor - 1)\}$ (i.e., at least $1/2$ of the initial population is filled up with k -means derived solutions). The resulting partitionings are then converted to MST-based genotypes, as illustrated in Fig. 8. The preservation of a high degree of MST information (within clusters) at this stage is crucial for the quick convergence of the algorithm. (If no MST information was used, the genetic variation within the population would be too high, making the crossover operator more disruptive.) Note that the numbers of clusters obtained as the final phenotypes are not predefined

and can increase or decrease depending on the structure of the underlying MST.

8) Main Loop of Phase I: The main loop of PESA-II is triggered by storing all of the *nondominated* solutions generated by initialization in the EP. PESA-II then runs as normal until the maximum number of generations is reached. At that point, the model selection phase of MOCK begins. The specific parameter settings recommended for MOCK and used in our experiments, including the maximum number of generations, are discussed in Section III-C.

B. Phase II: Model Selection

As noted previously, multiobjective clustering does not return a single solution, but a set of clustering solutions. These individual partitionings correspond to different tradeoffs between the two objectives and, in our case, also consist of different numbers of clusters. This means that the partitionings returned also have the opportunity to reflect the *hierarchical* structure of the data, where this exists (cf. Fig. 9), where different hierarchical levels for $k = 2$, $k = 3$, and $k = 4$ can be distinguished). Hierarchical information is very useful in many applications, as Duda and Hart point out: “... this kind of clustering [hierarchical] permeates classificatory activities in the science” [1, p. 229]. Moreover, the opportunity to analyze several alternative solutions means that any specialized domain expertise available can also be exploited, which is crucial for effective exploratory data analysis [3, Sec. 3.5].

On the other hand, other applications may require that the range of possible solutions should be narrow or that just one best solution be selected, automatically. In this section, we therefore introduce an automated method for assessing the quality of individual clustering solutions. This method can be used to identify (one or more) promising clustering solutions in the candidate set; the selection of a single solution then automatically delivers an estimate of the number of clusters inherent to the data set.

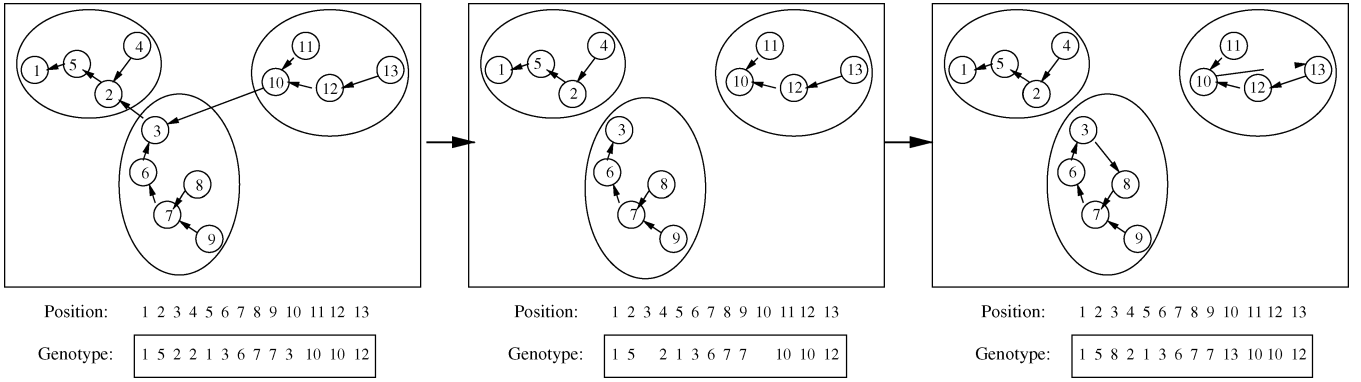


Fig. 8. Construction of an MST-similar solution from a given k -means solution. Starting from the original MST solution, all links that cross cluster boundaries (defined by the three-cluster k -means solution indicated here by the ellipses) are removed. Each deleted link (emanating from data item i) is then replaced by a link from i to a randomly chosen neighbor j with $j = nn_{il} \wedge l \leq L$. (Note that j need not be in the same k -means cluster as i .)

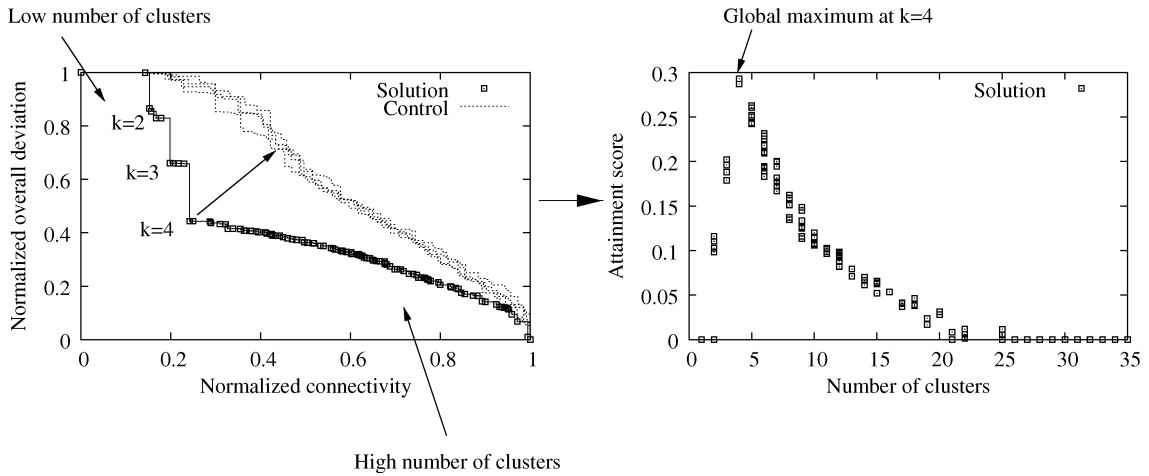


Fig. 9. On the left, solution and control fronts for a run of MOCK on the *Square1* data set (cf. Section IV-C) are shown. They are visualized as attainment surfaces (i.e., points are joined up with a stepped line). The solution with the largest minimum distance to the control fronts is indicated by the angled line, and corresponds to the desired $k = 4$ cluster solution. More generally, the distance of a solution from the nearest point on the control front is called its attainment score. On the right, attainment scores for the *Square1* data set are plotted as a function of k . The global maximum at $k = 4$ is clearly visible.

1) *Motivating Concepts*: The identification of promising solutions from Pareto front approximations has been investigated in several recent works [52]–[56]. These papers have generally dealt with the *reduction* of the size of the approximation set in the absence of additional knowledge about user preferences, and this is done to guide or focus the search towards the (potentially) more important areas. In contrast, we seek to first obtain the most complete Pareto front approximation set possible, and then to, *a posteriori*, reduce this set to a single solution. Thus, our method is quite different to those above, and the more so because we also make use of several domain-specific considerations that enable us to make a more effective technique for our particular purposes.

The approach we have developed is inspired by Tibshirani *et al.*'s Gap statistic [57], a statistical method to determine the number of clusters in a data set. The Gap statistic is based on the expectation that the most suitable number of clusters shows in a significant “knee” when plotting the performance of a clustering algorithm (in terms of a selected internal evaluation measure) as a function of the number of clusters. As internal evaluation measures are generally biased by the number of clusters (they show an increasing/decreasing trend that is solely due to a change in the number of clusters),

the “knee” can be identified best in a normalized plot, that is a performance plot that takes out the bias resulting purely from a change in the number of clusters. Tibshirani *et al.* realize this by generating a number of reference partitionings for random control data. From the normalized performance curve, they then identify the smallest number of clusters for which the gain in performance is not higher than would be expected for random data.

2) *Proposed Approach*: Like all existing methods for model selection,⁴ Tibshirani *et al.*'s Gap statistic [57] can be considered a single-objective approach to model selection, as it takes only a single clustering objective into account. Yet, it is our contention, that, in order to retain any advantages obtained through the use of a multiobjective clustering phase, the phase of model selection will itself need to make use of several separate objectives, which the approach proposed does.

Certain aspects of the Gap statistic can be carried over to the case of two objectives, as shown in the following. Fig. 9, which shows solutions plotted in the two-objective space, illustrates that the structure of the data is reflected in the shape

⁴This includes validation techniques such as the Silhouette Width or the Dunn Index, which can be considered as a fixed, and therefore ultimately single-objective, combination between two objectives.

of the Pareto front approximation set. From the two objectives employed, overall deviation decreases with an increasing number of clusters, whereas connectivity increases. Hence, generally, when considering two solutions with k and $k + 1$ clusters, respectively (where $k \in \{1, \dots, N\}$, and N is the size of the data set), we can say that we gain an improvement in overall deviation $\delta_{\text{overall_deviation}}$ at the cost of a degradation in connectivity $\delta_{\text{connectivity}}$. For k smaller than the true number of clusters (inherent to the data), we expect the ratio $R = \delta_{\text{overall_deviation}} / \delta_{\text{connectivity}}$ to be large: the separation of two clusters will trigger a great decrease in overall deviation, with only a small increase in connectivity (because individual clusters are being separated). When we surpass the correct number of clusters this ratio will diminish: the decrease in overall deviation will be less significant but come at a high cost in terms of connectivity (because a true cluster is being split). Using this knowledge, consider the plot of the Pareto front approximation in Fig. 9. Due to the natural bias of both measures, the solutions are approximately ordered by the number of clusters they contain: k gradually increases from left to right. The distinct change in R occurring for the correct number of clusters can therefore be seen as a “knee.” In order to help us correctly determine this knee, we use Tibshirani *et al.*’s concept of random control distributions, also referred to as a null model. Clustering a number of such distributions using MOCK, provides us with a set of “control fronts” (see Fig. 9).

3) *Control Data*: The purpose of the control data is to obtain an estimate of the values of connectivity and overall deviation that would be expected for unstructured data, i.e., data not containing clusters but covering the same data space (or manifold). The control data are obtained using a Poisson model in eigenspace as suggested in [58]. Specifically, a principal component analysis is applied to the covariance matrix of the original data. The eigenvectors and eigenvalues obtained are then used for the definition of a uniform distribution: the data is generated 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.

The Gap statistic performs normalization by, for each k , subtracting the mean expected objective value (computed on the control data), from the objective value obtained on the original data. Unfortunately, a normalization of our original “solution front” using the control fronts is not as straightforward. This is because both solution and control fronts contain not just one, but a set of different tradeoff solutions for every value of k , and it is therefore not clear how individual points in the solution front should be normalized. We overcome this problem by a heuristic approach, shown in Algorithm 1 and described next. Essentially, the approach proposed measures the distance between the fronts in two-objective space. For this purpose, the fronts first need to be aligned in objective space, such that they are directly comparable.

Algorithm 1 Identification of the Best Solution

- 1) **procedure** IDENTIFY
- 2) $SF :=$ solution front
- 3) **for** each i in 1 to #(control fronts) **do**

- 4) $CF_i := i$ th control front
- 5) **end for**
- 6) $k_{SF} :=$ largest k encountered in SF
- 7) **for** each i in 1 to #(control fronts) **do**
- 8) $k :=$ largest k encountered in CF_i
- 9) $k_{CF} := \min(k_{CF}, k)$
- 10) **end for**
- 11) $k_{\max} := \min(k_{\text{user}}, \min(k_{SF}, k_{CF}))$
- 12) $SF := \text{FILTER}(SF, k_{\max})$ /*(line 35)*/
- 13) **for** each i in 1 to #(control fronts) **do**
- 14) $CF_i := \text{FILTER}(CF_i, k_{\max})$
- 15) **end for**
- 16) $SF := \text{NORMALIZE}(SF)$ /*(line 39)*/
- 17) **for** each i in 1 to #(control fronts) **do**
- 18) $CF_i := \text{NORMALIZE}(CF_i)$
- 19) **end for**
- 20) $\text{best_solution} := -1$
- 21) $\text{min_dist} := \infty$
- 22) **for** each s in SF **do**
- 23) **for** each i in 1 to #(control fronts) **do**
- 24) **for** each $c \in CF_i$ **do**
- 25) $s_{\text{dist}} := \text{DISTANCE}(s, c)$ /*(line 44)*/
- 26) $\text{min_dist} := \min(\text{min_dist}, s_{\text{dist}})$
- 27) **if** $\text{min_dist} = s_{\text{dist}}$ **then**
- 28) $\text{best_solution} := s$
- 29) **end if**
- 30) **end for**
- 31) **end for**
- 32) **end for**
- 33) **return** best_solution
- 34) **end procedure**
- 35) **procedure** FILTER(solution set S , K)
- 36) remove all solutions with $k > K$ from S
- 37) **return** modified S
- 38) **end procedure**
- 39) **procedure** NORMALIZE(solution set S)
- 40) normalize the range of objective values encountered in S to $[0, 1]$
- 41) take the square root of each objective value in S
- 42) **return** modified S
- 43) **end procedure**
- 44) **procedure** DISTANCE(s, c)
- 45) **return** the shortest Euclidean distance between s and the attainment surface through c
- 46) **end procedure**

4) *Alignment of Solution and Control Fronts*: Given both solution and control fronts, we set $k_{\min} = 1$ and identify k_{\max} , the highest number of clusters shared by all fronts (lines 7–10). Alternatively, k_{\max} may also be limited by a user-specified parameter, k_{user} . Subsequently, we restrict the analysis to solutions with a number of clusters $k \in \{k_{\min}, \dots, k_{\max}\}$ (call of procedure FILTER in lines 12–15). For each front, we then determine the minimum and maximum value of both overall deviation and connectivity, and use these to scale all objective values to lie within the region $[0, 1] \times [0, 1]$. We further transform the objective values by taking the square root of each objective (call of procedure NORMALIZE in lines 16–19), a step motivated by the observation that both overall deviation and connectivity show a nonlinear development with respect to k . Overall deviation decreases very rapidly for the first few k ’s, while changes

TABLE I
PARAMETER SETTINGS FOR MOCK, WHERE N IS DATA SET SIZE

Parameter	setting
Number of generations	1000
External population size	1000
Internal population size	10
Resolution of hypergrid per dimension	10
Maximum number of clusters k_{user}	25 or 50
#(Initial solutions) $fsize$	$2 \times k_{user}$
Initialization	Minimum spanning tree and k -means ($L = 10$)
Mutation type	L nearest neighbours ($L = 10$)
Mutation rate p_m	$p_m = \frac{1}{N} + (\frac{L}{N})^2$
Recombination	Uniform crossover
Recombination rate p_c	0.7
Objective functions	Overall deviation and connectivity ($L = 10$)
#(Reference distributions)	3

for higher numbers of clusters are far less marked. Connectivity, in contrast, rises very quickly for higher numbers of clusters, while initial changes in the degree of connectivity are rather small. This results in an uneven sampling of the range of objective values, which—in a plot of the Pareto front—shows in a high density of points at the tails, with fewer solution points in the center. Taking the square root of the objective values is an attempt to reduce this “squeezing” effect, and give a higher degree of emphasis to small (but distinct) changes in the objectives. By this means, the algorithm becomes more accurate at identifying solutions situated in all parts of the solution front, in particular, those at the tails which may correspond, e.g., to partitionings with elongated cluster shapes or a high number of clusters.

5) *Attainment Scores:* For both solution and control fronts, we subsequently compute the attainment surfaces [59]. The attainment surface of an approximation set is uniquely defined by the points in the set. It is the boundary in the objective space separating those points that are dominated by or equal to at least one of the data points in the set from those that no data point in the set dominates or equals (see Fig. 9). In lines 20–32 we compute, for each solution point p , its “attainment score” as the Euclidean distance between p and the closest point on any of the reference attainment surfaces (call of procedure DISTANCE in line 25).

Finally, we plot the attainment scores as a function of the number of clusters k (see Fig. 9). All solutions corresponding to the local optima in the resulting plot are considered as promising solutions. The global maximum in this plot may be considered as the estimated “best” solution (line 33).

C. Parameter Settings

The parameter settings used for MOCK in our experimental study are given in Table I. Apart from the maximum number of clusters k_{user} , these parameters are kept constant over the entire range of data sets in our comparison study.

The majority of MOCK’s parameters are just those of PESA-II. In our experiments, a size limit of 1000 for the EP proves to be more than sufficient to store all nondominated solutions; with all other parameter settings fixed, the final number of nondominated solutions returned is usually only around 100. Clearly, the accuracy of the algorithm may be improved by increasing the size of the IP or the number of iterations. Our

choices for mutation and crossover rates, the resolution of the hypergrid, and any other parameters used within PESA-II are standard values from the literature.

Selection of the number of control distributions requires a compromise between accuracy and computational cost. Evidently, additional control surfaces result in a smoother control surface and therefore in a slightly more accurate estimate of solution desirability. However, every control surface comes at a substantial additional cost in running time, and we therefore keep the number of controls relatively low. If the tradeoff curve is explored visually, even a single control distribution may be sufficient.

The value of k_{user} can be interpreted as an upper bound on the number of clusters expected in the data set. Its is not crucial to set this precisely, hence quite large values can be used as a default. However, smaller values will yield more accurate model selection results when the data actually has few clusters. For example, in our experiments in Section V, we have used the value 50 for the randomly generated data sets with up to 40 (but as few as 4) clusters and a value of 25 for the handcrafted data sets, which have at most six clusters.

The definition of the parameter L , used during initialization, mutation, and in the definition of connectivity, determines the algorithm’s sensitivity to small clusters. While the choice of a reasonably large value of L is necessary to prevent outliers from being classified as individual clusters, a choice that is too large may result in clusters being overlooked, if these are very small (e.g., if they contain less than $L/2$ data items). In our experiments, we have chosen a value of $L = 10$, which still permits the robust detection of clusters consisting of very few data items, and has resulted in an adequate performance across the range of data sets studied. In general, we recommend to set this parameter within the range $L \in \{5, \dots, 20\}$.

IV. EXPERIMENTAL SETUP

A. Contestant Methods

In our experiments, we aim to show that the conceptual advantage of multiobjective clustering translates into an actual practical performance advantage compared with algorithms based on similar clustering criteria. For this purpose, we compare MOCK against a set of different algorithms.

1) Three well-known and conceptually different single-objective clustering algorithms: the partitioning method k -means [16], and two agglomerative hierarchical methods [17] based on the linkage criteria of single link and average link. Note that, with regard to these three methods, we intentionally choose a set of conceptually simple algorithms. This is because our aim is to compare MOCK to clustering algorithms optimizing similar clustering objectives to MOCK, albeit only one at a time. This way we can demonstrate that MOCK achieves a high level of performance not because the objectives it optimizes are new or different, but rather, because it uses more than one of them simultaneously. Thus, k -means and single link agglomerative clustering, in particular, provide a most relevant baseline because each uses a clustering objective that is conceptually very similar to one of MOCK’s (overall deviation and connectivity, respectively), and is highly efficient at optimizing this particular objective. In this context, it is fundamental to understand

that the single-objective algorithms fail on particular data sets not because they fail to optimize their objective function (they usually beat MOCK in this respect, see Fig. 6), but because they set out optimizing the wrong objective in the first place. Average link can also be seen as an optimizer of cluster compactness, and is included in our study due to its popularity in the data-mining literature (e.g., in bioinformatics and text-mining).

2) A cluster ensemble integrating the solutions returned by k -means, average link and single link. As outlined in the introduction, the principal ideas underlying cluster ensembles are related to those of multiobjective clustering. By comparing to a cluster ensemble technique, we aim to establish whether there is an additional benefit to optimizing different objectives *during clustering*, as opposed to the *retrospective combination of partitions* that is the basis of ensemble methods. We chose the ensemble technique proposed in [9], a choice motivated by its high profile and popularity in the literature and recent results [36] suggesting that graph-based approaches (of which this is one) may be more robust towards varying data properties than co-association-based techniques.

Finally, we consider two of the most popular methods for the determination of the number of clusters in a data set: the Gap statistic [57] and the Silhouette Width [60]. Comparison to the Gap statistic is interesting, as it shares certain concepts with our method of solution selection, in particular, the use of random control data. On the other hand, comparison to the Silhouette Width is interesting, as this method takes two objectives (combined in a fixed nonlinear combination) into account.

All of the above methods are described in more detail next.

1) *Obtaining Comparable Sets of Solutions*: Of course, MOCK generates a range of solutions with different numbers of clusters in an individual run, whereas the single-objective algorithms and the clustering ensemble generate only a single solution for a prespecified number of clusters. Thus, in order to obtain a set of solutions comparable to the set returned by MOCK, k -means and the clustering ensemble are each run for a range of different numbers of clusters (identical to the range covered by MOCK). For the agglomerative algorithms, single link and average link, the set of solutions is obtained by cutting the returned dendrogram at different levels.

Note that generating such a range of clustering solutions is preferable to providing the algorithms with the correct number of clusters: even if the actual number of clusters in the data is known, the best number of clusters for the different algorithms may vary (and not be known *a priori*). This is predominantly because some algorithms (e.g., single link) tend to isolate outliers into individual clusters. The best number of clusters for a specific algorithm may thus be quite different to the known “correct” number of clusters.

2) *k-Means*: We use a standard implementation of the k -means algorithm [16], based on the batch version where cluster centers are only recomputed after the reassignment of all data items. To reduce suboptimal solutions, k -means is run repeatedly (ten times) using random initialization (which is known to be an effective initialization method [61]) and only the best result in terms of intracluster variance is returned.

3) *Hierarchical Clustering*: As a second and third method, two hierarchical clustering algorithms, single link agglomera-

tive clustering and average link agglomerative clustering, are implemented [17]. Both of these algorithms are deterministic.

4) *Cluster Ensemble*: Strehl and Ghosh’s ensemble technique employs three conceptually different ensemble methods, namely: 1) cluster-based similarity partitioning algorithm (CSPA); 2) hypergraph partitioning algorithm (HGA); and 3) metaclustering algorithm (MCLA) [9]. The solutions returned by the individual combiners then serve as the input to a supra-consensus function, which selects the best solution in terms of average shared mutual information.

For our experiments, we use Strehl and Ghosh’s original MATLAB code (available from [62]). The input labels are generated using k -means, average link and single link hierarchical clustering. As ensemble methods generally benefit from being provided partitionings of higher resolution (i.e., comprising more clusters), we run each algorithm for a range of different numbers of clusters $\{1, \dots, k_{\text{user}}\}$.⁵ The resulting set of labellings then serves as the input to Strehl and Ghosh’s method.

5) *Gap Statistic*: To compute the Gap statistic [57], the clustering problem is solved for a range of different values of k and, for each k , the resulting partitioning $C = \{C_1, \dots, C_k\}$ is evaluated by means of the intracluster variance, which is given by

$$V(C) = \sum_{C_l \in C} \sum_{i \in C_l} (\delta(i, \mu_l))^2. \quad (8)$$

Here, C_l is the l th cluster in the partitioning, μ_l is the corresponding cluster center, and $\delta(i, \mu_l)$ gives the dissimilarity between data item i and μ_l . The intracluster variance is affected by the number of clusters, such that a plot $\text{Var}(k)$ showing the evolution of $V(C)$ as a function of the input parameter k exhibits a decreasing trend that is solely caused by the finer partitioning and not by the actual capturing of structure within the data. The Gap statistic overcomes this effect through a normalization of the performance curve. B reference curves $R_b(k)$ (with $b \in \{1, \dots, B\}$) are computed, which are the performance curves obtained with the same clustering algorithm for uniform random reference distributions. Using these, the normalized performance curve (“Gap curve”) for $\text{Var}(k)$ is given as

$$\text{Gap}(k) = \frac{1}{B} \sum_{b=1}^B \log(R_b(k)) - \log(\text{Var}(k)). \quad (9)$$

The most suitable number of clusters is then determined as the smallest k , such that

$$\text{Gap}(k) \geq \text{Gap}(k+1) - s_{k+1} \quad \text{where} \quad s_k = sd_k \sqrt{1 + \frac{1}{B}} \quad (10)$$

and sd_k is the standard deviation within the set of the B log-scaled reference points $R_b(k)$.

⁵While in [9] the correct value of k only is used, results presented by the same authors in [63] indicate that the use of a range of different numbers of clusters yields improved solution quality. Experiments on our own data sets confirm this.

For our implementation of the Gap statistic, we use the k -means algorithm for cluster generation and a Poisson distribution in eigenspace for the null model (as suggested in [57], and as used in MOCK). We compute the performance curves for a range of different numbers of clusters, and, for each k , we generate $B = 10$ reference distributions.

6) *Silhouette Width*: The Silhouette Width [60] is a popular, general-purpose validation technique commonly used for model selection in cluster analyses. Notably, it can be considered a (fixed) nonlinear combination of two objectives, as it takes both intracluster distances and intercluster distances into account. The procedure for its use in model selection is as follows: for each k in a given range, the Silhouette Width of the corresponding clustering solution is computed. In a plot of the Silhouette Width as a function of the number of clusters, the global maximum is identified and considered the estimated best solution.

The Silhouette Width for a partitioning is computed as the average Silhouette value over all data items. Here, the Silhouette value for an individual data item i , which reflects the confidence in this particular cluster assignment, is computed as

$$S(i) = \frac{b_i - a_i}{\max(b_i, a_i)} \quad (11)$$

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 returns values in the interval $[-1, 1]$ and is to be maximized.

B. Evaluation of Clustering Quality

For the reasons discussed in Section I, clustering quality is evaluated using an external measure of clustering quality, which can be considered an objective evaluation. Specifically, we choose the Adjusted Rand Index [64], which is a generalization of the Rand Index. The Rand indices take two partitionings as the input (one of which, in our case, is the known correct solution) and count the number of pairwise co-assignments of data items between the two partitionings. The Adjusted Rand Index additionally introduces a statistically induced normalization in order to yield values close to 0 for random partitions. This normalization is particularly important in our experiments, as we compare partitions containing different numbers of clusters, which may otherwise affect the ordering of results [8].

Using a representation based on contingency tables, the Adjusted Rand Index [65] is given as

$$R(U, V) = \frac{\sum_{lk} \binom{n_{lk}}{2} - [\sum_l \binom{n_{l\cdot}}{2} \cdot \sum_k \binom{n_{\cdot k}}{2}] / \binom{n}{2}}{\frac{1}{2} [\sum_l \binom{n_{l\cdot}}{2} + \sum_k \binom{n_{\cdot k}}{2}] - [\sum_l \binom{n_{l\cdot}}{2} \cdot \sum_k \binom{n_{\cdot k}}{2}] / \binom{n}{2}} \quad (12)$$

where n_{lk} denotes the number of data items that have been assigned to both cluster l and cluster k . The Adjusted Rand Index return values in the interval $[\sim 0, 1]$ and is to be maximized.

TABLE II

RANDOMLY GENERATED DATA SETS. THE GAUSSIAN CLUSTER GENERATOR WAS USED FOR LOW DIMENSIONALITIES (2 AND 10), AND THE ELLIPSOID CLUSTER GENERATOR WAS USED FOR HIGH DIMENSIONALITIES (50 AND 100). FOR EACH OF THE 16 COMBINATIONS OF CLUSTER NUMBER AND DIMENSION, TEN DIFFERENT INSTANCES WERE GENERATED, GIVING 160 DATA SETS IN ALL. IN OUR EXPERIMENTS, THESE GROUPS OF TEN INSTANCES ARE REFERRED TO AS x d - y c, WHERE x IS THE DIMENSIONALITY AND y IS THE NUMBER OF CLUSTERS IN THE DATA SET

Parameter	range
Number of clusters	4, 10, 20, 40
Dimension	2, 10, 50, 100
Size of each cluster	uniformly in $\{50, \dots, 500\}$ for 4 and 10 cluster instances, and $\{10, \dots, 100\}$ for 20 and 40 cluster instances

C. Experimental Data

Two different groups of experimental data are used in this study. These data sets and the corresponding generators are available from <http://www.dbk.ch.umist.ac.uk/handle/generators/>.

1) *Handcrafted Data*: The first group of data consists of a suite of 21 handcrafted two-dimensional (2-D) data sets, which are used to study a range of different interesting data properties (see Fig. 10). In particular, we are interested in the algorithms' robustness to features such as overlap between clusters, unequally sized clusters, and elongated cluster shapes. The clusters in these data sets are described by 2-D normal distributions $N(\vec{\mu}, \vec{\sigma})$ with a fixed cluster size, a fixed mean vector $\vec{\mu}$ and a fixed vector of the standard deviation $\vec{\sigma}$, except for some of the clusters in the Spiral, Smile, and Spiralsquare data sets, which are described geometrically. In each run of the experiments, a new instance of size 1000 is sampled from these distributions.

2) *Randomly Generated Data*: More general test data (up to 100-dimensional) were obtained using the generators described in [14]. The first of these generators is based on a standard cluster model using multivariate normal distributions. Here, clusters are generated based on a simple trial-and-error scheme: they are iteratively constructed, and a simple heuristic is used to detect overlap between them. Newly introduced clusters with overlap are rejected and regenerated, until a valid set of clusters has been found.

In low dimensions, the clusters obtained using this generator are frequently elongated and of arbitrary orientation (see Fig. 10, data set 2d-40c). However, in higher dimensions, 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 nondiagonal 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, a second cluster generator producing more elongated cluster shapes in arbitrarily high dimensions was used.

This second generator creates ellipsoidal clusters with the major axis at an arbitrary orientation. A genetic algorithm is used to translate the location of the origin of each cluster so that a cost consisting of the compactness of the entire data set, plus a penalty term for any overlapping clusters is minimized.

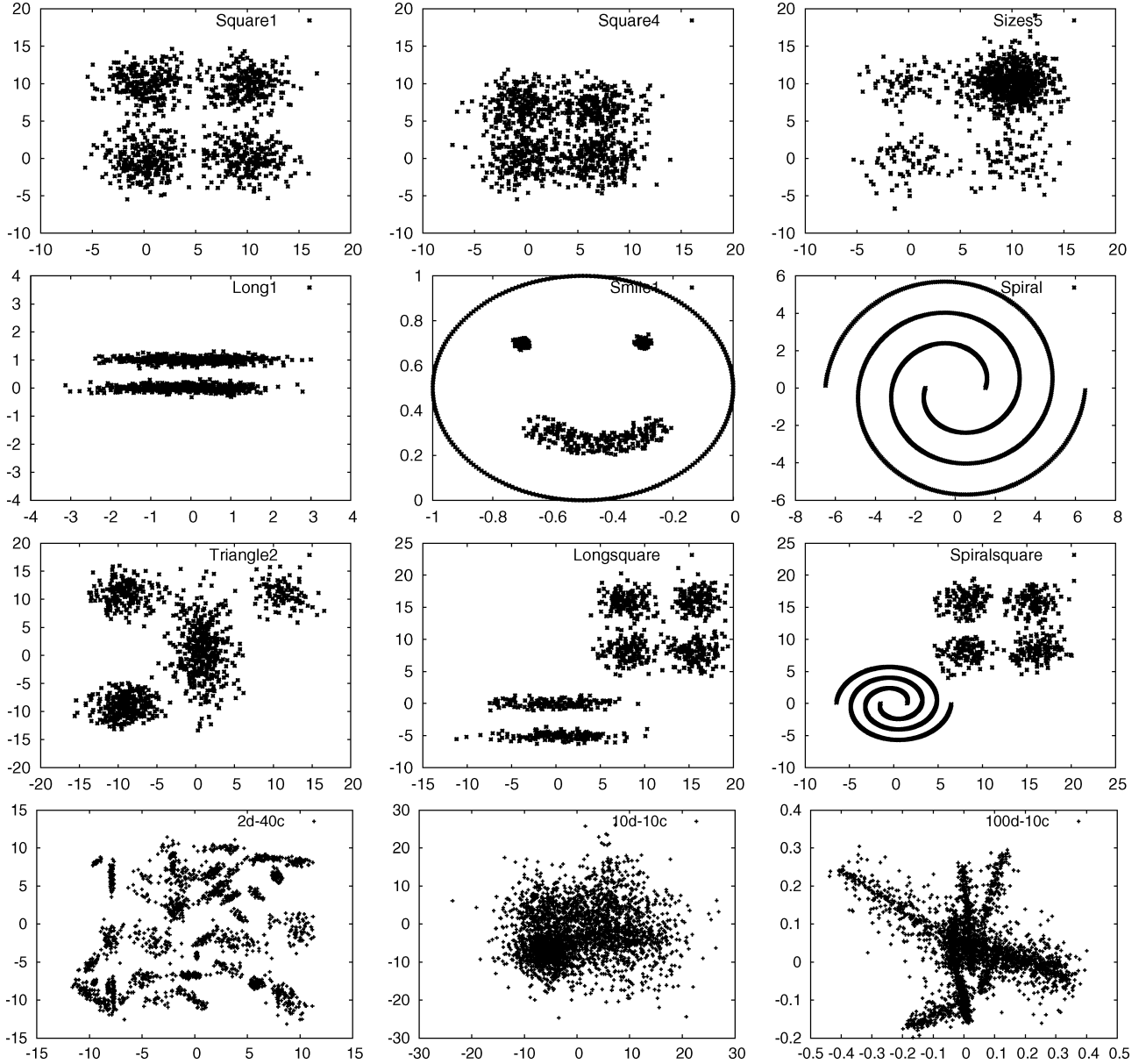


Fig. 10. Sample instances of the synthetic data sets employed. The first nine figures (from left to right, top to bottom) are examples of handcrafted data sets. *Square1* consists of four clusters of equal size and spread. The *Square* series and the *Sizes* series are variations of this data set, that vary the degree of overlap and the relative size of clusters, respectively. Due to the lack of spatial separation between clusters, these problems are hard to solve for algorithms based on optimizing connectedness or spatial separation. The *Long* series, *Smile* series, *Triangle* series, and the *Spiral* data set contain elongated cluster shapes that are hard to identify for algorithms based on cluster compactness. Finally, *Longsquare* and *Spiralsquare* combine different types of clusters, and are therefore hard to solve for both methods based on cluster compactness and those based on connectedness. The last three figures are examples of randomly generated data sets. 2d-40c is a 2-D data set containing 40 Gaussian clusters. 10d-10c and 100d-10c are ten-cluster data sets of 10 and 100 dimensions, respectively (here, their projections in just two dimensions are shown).

The 160 data sets obtained from these generators are summarized in Table II, and three sample data sets are shown in Fig. 10.

D. Distance Function

Distance computation for all data sets is done using the Euclidean distance (in the unnormalized data space).

V. RESULTS

In the following section, we discuss the results obtained in our experiments. Specifically, we will investigate the effective-

ness of both phases of the MOCK algorithm, that of 1) cluster generation and 2) model selection.

A. Evaluation of the Clustering Phase

The aim of this part of the assessment is to determine the algorithms' ability to *generate* high-quality partitionings. For each algorithm, once all solutions (corresponding to different numbers of clusters) for a single "run" have been generated, the highest Adjusted Rand Index value from amongst this set of solutions is determined. Thus, we are using external knowledge to select the best solution generated by each algorithm but no

TABLE III
NUMBER OF CLUSTERS AND QUALITY OF THE BEST SOLUTION (IN TERMS OF THE ADJUSTED RAND INDEX, AND WITHIN THE CLUSTER RANGE OF $k \in \{1, \dots, 25\}$) GENERATED BY (FROM LEFT TO RIGHT) MOCK, k -MEANS, AVERAGE LINK, SINGLE LINK, AND THE CLUSTER ENSEMBLE ON A NUMBER OF ARTIFICIAL DATA SETS EXHIBITING DIFFERENT CLUSTER PROPERTIES (MEAN VALUES FROM 50 RUNS). THE STATISTICALLY BEST ($\alpha = 0.005$) PERFORMER(S) ARE HIGHLIGHTED IN BOLD FACE. TO TEST IF DIFFERENCES WERE STATISTICALLY SIGNIFICANT, A KRUSKAL-WALLIS RANK TEST [66] WAS APPLIED TO ALL PAIRS OF ALGORITHMS

Problem		MOCK		k -means		Av. link		Single link		Ensemble	
Name	k	k	Rand	k	Rand	k	Rand	k	Rand	k	Rand
Square1	4	4.22	0.9622	4	0.9651	4.16	0.9370	19.98	0.3798	4	0.9626
Square2	4	4.32	0.9262	4	0.9358	4.24	0.8971	18.96	0.0876	4	0.9318
Square3	4	4.32	0.8643	4	0.8844	4.12	0.8293	20.7	0.0058	4	0.8807
Square4	4	4.32	0.7729	4	0.8048	4.44	0.7320	20.26	0.0002	4.04	0.7865
Square5	4	4.74	0.6304	4	0.6814	4.98	0.5915	19.86	0.0002	4	0.6716
Sizes1	4	4.28	0.9639	4	0.9666	4.32	0.9412	20.84	0.4603	4.9	0.6936
Sizes2	4	4.2	0.9697	4	0.9654	4.28	0.9510	22.18	0.5629	2	0.4890
Sizes3	4	4.2	0.9748	4	0.9633	4.26	0.9601	21.3	0.6293	2.52	0.3133
Sizes4	4	4.24	0.9763	4	0.9551	4.4	0.9581	22.1	0.7067	3.26	0.2305
Sizes5	4	4.2	0.9760	3.92	0.9557	4.32	0.9581	21.92	0.7933	3.6	0.1656
Triangle1	4	4	1	4	0.9582	4.82	0.9926	4.14	0.9998	4	0.9974
Triangle2	4	4.1	0.9864	4	0.8858	5.02	0.9543	16.58	0.8096	3.72	0.5817
Long1	2	2	0.9998	4.98	0.3562	7.64	0.4717	3.48	0.9957	2.18	0.9000
Long2	2	2	0.9999	4.94	0.3388	7.84	0.4572	3.42	0.9959	2.4	0.5287
Long3	2	2	0.9996	4.44	0.2168	7.32	0.3785	3.32	0.9961	5.54	0.0914
Smile1	4	4	1	20.14	0.7396	11.98	0.8008	4	1	3.96	0.6581
Smile2	4	4	1	10.36	0.5502	10.54	0.7561	4	1	4.94	0.5727
Smile3	4	4	1	19.88	0.3428	11.22	0.3826	4	1	4.12	0.3562
Spiral	2	2	1	5.12	0.0515	25	0.0818	2	1	4.8	0.0377
Longsquare	6	6.02	0.9844	7.02	0.8549	9.26	0.9052	19.22	0.7924	6.1	0.9390
Spiralsquare	6	6.06	0.9970	2.86	0.4864	5	0.5386	17.06	0.9843	7.62	0.0588

external knowledge is used by the algorithms themselves. This testing procedure is defined more precisely in Algorithm 2.

Algorithm 2 Cluster Generation Testing Procedure

- 1) **for** each problem p **do**
 - 2) **for** each algorithm a **do**
 - 3) **for** each run r **do**
 - 4) Run a on p to generate a range of partitionings with different numbers of clusters in $\{1, \dots, k_{\text{user}}\}$. For MOCK this is done implicitly in a single execution; k -means and the ensemble have to be executed for each different k ; for the agglomerative algorithms the dendrogram generated is cut at a range of different levels.
 - 5) Compute the Adjusted Rand Index of all the solutions generated.
 - 6) Store the best Adjusted Rand Index value.
 - 7) **end for**
 - 8) Compute the mean of the best Adjusted Rand Index for algorithm a on problem p .
 - 9) Display the mean in Table III or Table IV.
 - 10) Use the distribution of values for statistical tests.
 - 11) **end for**
 - 12) **end for**
-

1) *Handcrafted Data*: Given the algorithms chosen and the cluster properties studied in these data, certain results can be expected in our experiments. First, a strong performance of the k -means algorithm and average link on the Square and Sizes data sets can be expected, as these contain spherical Gaussian

clusters, which is the cluster model assumed by both methods. In contrast, we would expect a poor performance of these algorithms on data sets containing clusters of elongated shapes (in particular, Spiral, Smile, and Long; see Fig. 10). Second, a strong performance of single link on the Spiral, Smile, and Long data sets can be expected, as these data sets contain spatially well-separated clusters, which is the main assumption made by the single link linkage criterion. Different to k -means, this algorithm makes no assumption on the shape of the clusters, so clusters of arbitrary shape can be detected. We would, however, expect a poor performance for those data sets displaying significant amounts of cluster overlap (in particular, Square and Sizes; see Fig. 10).

The results in Table III confirm these expectations. We can see that the individual single-objective methods give very good results on those data sets that comply with the assumptions made by their particular clustering criterion. However, they do not show a consistent performance across the range of different properties. In particular, k -means and average link tend to subdivide elongated cluster shapes, and single link tends to isolate outliers on data sets with overlap. In Table III, this is reflected in low values of the Adjusted Rand Index and in a significant increase in the number of clusters corresponding to the best solution. The clustering ensemble method can improve on the performance of the individual single-objective algorithms on some data sets only. Generally, its performance breaks down drastically on those data sets containing unequally sized clusters (e.g., Sizes, Triangle).⁶

On the other hand, the results for MOCK display the increased robustness expected from the simultaneous optimization of

⁶This major limitation of the ensemble method is due to its use of graph partitioning techniques, which rely on the use of constraints concerning the balancedness of cluster sizes.

TABLE IV
NUMBER OF CLUSTERS k AND QUALITY OF THE BEST SOLUTION (IN TERMS OF THE ADJUSTED RAND INDEX, AND WITHIN THE CLUSTER RANGE OF $k \in \{1, \dots, 50\}$) GENERATED BY (FROM LEFT TO RIGHT) MOCK, k -MEANS, AVERAGE LINK, SINGLE LINK, AND THE CLUSTER ENSEMBLE. VALUES FOR MOCK, k -MEANS AND THE CLUSTER ENSEMBLE ARE MEAN VALUES FROM 21×10 RUNS; AVERAGE LINK AND SINGLE LINK ARE DETERMINISTIC, SO AVERAGES ARE OVER THE TEN DIFFERENT PROBLEM INSTANCES ONLY. THE STATISTICALLY BEST PERFORMER ($\alpha = 0.005$) IS HIGHLIGHTED IN BOLD FACE. TO TEST IF DIFFERENCES WERE STATISTICALLY SIGNIFICANT, A PAIRED WILCOXON SIGNED-RANK TEST [66], WHICH TAKES INTO ACCOUNT THE DEPENDENCE ON PROBLEM INSTANCE, WAS APPLIED TO ALL PAIRS OF ALGORITHMS

Problem	MOCK		k -means		Av. link		Single link		Ensemble	
	k	Rand	k	Rand	k	Rand	k	Rand	k	Rand
2d-4c	4.12	0.9893	3.99	0.9143	5	0.9330	18.2	0.8396	4.92	0.4669
10d-4c	4.07	0.9962	3.99	0.9704	7.7	0.9797	33.9	0.3965	10.31	0.4314
50d-4c	4	1	4.9	0.5616	15.6	0.6687	26.1	0.1349	28.01	0.1537
100d-4c	4	1	6.26	0.4941	11.8	0.6005	10.4	0.0125	32.56	0.1748
2d-10c	10.33	0.9451	11.85	0.8299	22	0.8329	36.9	0.6347	33.6	0.4193
10d-10c	12.68	0.9673	9.25	0.9250	29.3	0.9387	45.2	0.0027	10.89	0.3937
50d-10c	10	0.9990	18.11	0.5115	36	0.5165	29.1	0.0013	21.42	0.2789
100d-10c	10	0.9999	16.79	0.5152	30.5	0.5821	22.9	0.0013	19.43	0.2649
2d-20c	19.94	0.9454	33.79	0.8633	26.1	0.9369	40	0.8227	44.64	0.0425
10d-20c	20.26	0.9981	21.45	0.9820	20.2	0.9974	40.9	0.9463	29.26	0.2377
50d-20c	21.25	0.9146	36.14	0.5576	43.4	0.4873	41.6	0.0066	19.86	0.7288
100d-20c	20.12	0.9027	34.1	0.5686	46.1	0.5590	38	0.0167	14.31	0.7022
2d-40c	42.14	0.8654	42.36	0.6932	44.8	0.8231	47	0.5324	23.45	0.1181
10d-40c	42.84	0.9896	43.48	0.9678	42.3	0.9957	44.8	0.7454	22.15	0.2469
50d-40c	46.57	0.8216	47.4	0.4671	48.9	0.2007	41.6	0.0016	22.38	0.3904
100d-40c	45.99	0.7988	48.12	0.4747	48.3	0.2288	48.1	0.0028	24.34	0.4108

both clustering objectives. While it may be marginally beaten by single-objective methods on data sets those are perfectly suited for (here, k -means on the Square data sets), it shows an impressive performance across the entire range of data properties. This is not only reflected in the high values of the Adjusted Rand Index, but also in the close agreement between the number of clusters in the best generated solution, and the correct k on all data sets. Interestingly, the use of two objectives already pays off for data such as the Triangle data sets, which deviate only marginally from the spherical cluster model assumed by k -means (the Triangle data sets contains four Gaussian clusters, only three of which are spherical and one of which is elliptical).

2) *Randomly Generated Data*: To a certain degree, the handcrafted data studied above are artificial in that they contain geometric cluster shapes unlikely to occur in real applications, clusters are arranged in a regular fashion, and individual variables are uncorrelated. Using the second group of benchmark data, we therefore investigate whether MOCK's advantages on this "simple" handcrafted data carry over to more realistic, general types of data, high-dimensional data, and data sets with large numbers of clusters.

Table IV shows the results obtained for these randomly generated data sets. Here, MOCK is the sole best performer on 15 out of 16 of the groups of data sets, and is beaten on only one of them (and by a very small margin). In contrast, both k -means and average link have significant problems in dealing with the ellipsoidal clusters in 50 and 100 dimensions. Single link shows a very bad performance on almost all of these data sets, which is owing to a lack of clear spatial separation between clusters. The cluster ensemble technique generally performs very poorly, too.

B. Evaluation of the Model Selection Phase

We have seen above that multiobjective clustering indeed results in a performance advantage when considering the al-

gorithm's capability to generate high-quality solutions. Yet, in order to benefit from this advantage in practice, the use of an effective technique for model selection is essential. In this section, we seek to evaluate the performance of MOCK's method of solution selection. Specifically, we address the following issues.

- How does the clustering quality of the individual algorithms compare, if solutions selected by an appropriate method of model selection are taken into account only, rather than solutions selected by external knowledge?
- Does MOCK's attainment score method work better than another method of solution selection applied to the same set of solutions (i.e., the output of MOCK's clustering phase)?

For this purpose, we compare MOCK's attainment score method with the Silhouette Width and the Gap statistic. Specifically, we compare the solution returned after the second phase of MOCK with solutions selected by the Silhouette Width (when it is applied to each of the clustering algorithms, including MOCK's clustering phase). In addition, we also compare with the k -means solution picked by means of the Gap statistic. This testing procedure is defined more precisely in Algorithm 3.

Algorithm 3 Model Selection Testing Procedure

- 1) **for** each problem p **do**
- 2) **for** each algorithm a **do**
- 3) **for** each run r **do**
- 4) Run a on p to generate a range of partitionings with different numbers of clusters in $\{1, \dots, k_{\text{user}}\}$. For MOCK this is done implicitly in a single execution; k -means and the ensemble have to be executed for each different k ; for the agglomerative algorithms the dendrogram generated is cut at a range of different levels.

TABLE V

NUMBER OF CLUSTERS k AND QUALITY OF THE SOLUTION (IN TERMS OF THE ADJUSTED RAND INDEX, AND WITHIN THE CLUSTER RANGE OF $k \in \{1, \dots, 25\}$) DETERMINED BY (FROM LEFT TO RIGHT) MOCK'S ATTAINMENT SCORE METHOD, THE SILHOUETTE WIDTH (APPLIED TO THE OUTPUT OF MOCK, k -MEANS, AVERAGE LINK, SINGLE LINK, AND THE CLUSTER ENSEMBLE), AND THE GAP STATISTIC, ON A NUMBER OF ARTIFICIAL DATA SETS EXHIBITING DIFFERENT CLUSTER PROPERTIES (MEAN VALUES FROM 50 RUNS). THE STATISTICALLY BEST ($\alpha = 0.005$) PERFORMER(S) ARE HIGHLIGHTED IN BOLD FACE. TO TEST IF DIFFERENCES WERE STATISTICALLY SIGNIFICANT, A KRUSKAL-WALLIS RANK TEST [66] WAS APPLIED TO THE SET OF ALGORITHMS

Problem		MOCK (att)		MOCK (sil)		k -means (sil)		Av. link (sil)		Single link (sil)		Ensemble (sil)		k -means (gap)	
Name	k	k	Rand	k	Rand	k	Rand	k	Rand	k	Rand	k	Rand	k	Rand
Square1	4	4	0.9553	4	0.9605	4	0.9651	4	0.9367	2.72	0.0374	4	0.9626	4	0.9651
Square2	4	4	0.9192	4	0.9236	4	0.9358	4	0.8966	2	8e-07	4	0.9318	3.8	0.8901
Square3	4	4.06	0.8506	3.98	0.8573	4	0.8844	4.06	0.8292	2	1e-06	4	0.8807	3	0.6323
Square4	4	4.32	0.7468	4	0.7677	4	0.8048	4.26	0.7032	2	9e-07	4.04	0.7865	1.16	0.0588
Square5	4	6.62	0.5215	3.96	0.6117	4	0.6814	3.66	0.4294	2	1e-06	4	0.6716	1	0
Sizes1	4	4.02	0.9592	4.02	0.9620	4	0.9666	4	0.9408	2.6	0.0380	4.34	0.6660	1	0
Sizes2	4	4	0.9654	4	0.9682	4	0.9654	4	0.9509	2.18	0.0210	2.3	0.4834	1	0
Sizes3	4	3.96	0.9637	4	0.9736	4	0.9633	4	0.9599	2.08	0.0235	2.64	0.2859	1	0
Sizes4	4	3.84	0.9379	4	0.9740	4	0.9551	4	0.9578	2.74	0.1318	2.86	0.1425	1	0
Sizes5	4	3.52	0.8725	3.88	0.9577	3.74	0.9453	3.76	0.9271	2.44	0.1027	3.7	0.0791	1	0
Triangle1	4	4	1	4	1	4	0.9582	4	0.9850	4.14	0.9998	4	0.9974	1	0
Triangle2	4	4.14	0.9561	4	0.9764	4	0.8858	4.06	0.9423	4.62	0.2795	2.3	0.3851	1	0
Long1	2	2.78	0.9462	8.34	0.3001	8.32	0.2761	7.78	0.1763	2.02	0.3000	4.92	0.3872	1	0
Long2	2	2.2	0.9786	7.98	0.2980	8.52	0.2566	6.7	0.1254	2.06	0.2001	4.82	0.2677	1	0
Long3	2	2.58	0.9690	7.16	0.2258	6.74	0.1844	5.84	0.0826	2.06	0.2636	3.1	0.0498	1	0
Smile1	4	4	1	13.98	0.7307	21.3	0.7380	17.42	0.7233	4	1	3.06	0.5864	2.96	0.5874
Smile2	4	4	1	14.1	0.5262	20.28	0.5408	18.02	0.5258	4	1	2	0.4391	1	0
Smile3	4	4	1	15.4	0.3247	20.9	0.3361	18	0.3043	3.6	0.8822	2.02	0.2082	1.52	0.0694
Spiral	2	3.16	0.9189	23.82	0.0966	19.68	0.0421	25	0.0818	2	1	3.76	0.0297	1	0
Longsquare	6	5.34	0.6665	3.14	0.3663	2.06	0.2843	2	0.2741	2.46	0.2715	5.58	0.9024	1	0
Spiralsquare	6	3.02	0.7697	23.6	0.2857	2	0.4167	2	0.4202	2.94	0.4250	3.14	0.0138	1.02	0.0083

- 5) Use the Silhouette Width, Gap statistic or the attainment score method (as appropriate) to select the estimated "best" solution.
- 6) Store the Adjusted Rand Index value of the estimated "best" solution.
- 7) **end for**
- 8) Compute the mean of the Adjusted Rand Index for algorithm a on problem p .
- 9) Display the mean in Table V or Table VI.
- 10) Use the distribution of values for statistical tests.
- 11) **end for**
- 12) **end for**

1) *Handcrafted Data:* Given the algorithms chosen and the cluster properties studied in these data, certain results can again be expected in our experiments. In particular, a strong performance of the Silhouette Width on all data sets containing spherically shaped clusters is expected, as the Silhouette Width aims to maximize both intracluster homogeneity and inter-cluster heterogeneity. Due to shared modelling assumptions, the Silhouette Width can also be expected to perform particularly well in combination with k -means and average link.

Again, the above expectations are reflected in the experimental results, which are shown in Table V. In particular, the Silhouette Width shows a very strong performance on the Square and Sizes data sets, both in the selection of k -means' and of MOCK's clustering results. Yet, this good performance does not carry over to most data sets containing elongated cluster shapes (even though, somewhat surprisingly, the Silhouette Width manages to identify good single link solutions for the Smile and Spiral data sets). Similarly, the Gap statistic performs poorly for elongated cluster shapes. Moreover, it does

not cope with an absence of clear spatial separation between clusters.

The most striking result, however, is that MOCK (using its own attainment score method of model selection) performs the most robustly (of all the combinations of clustering plus model-selection methods) across the range of different data sets. On those data sets containing spherically shaped clusters, it does not perform quite as well as the Silhouette Width, but its performance is still high and extends to the Long and Smile data sets. Its performance breaks down slightly only for the Spiralsquare and Longsquare data sets, where it underestimates the number of clusters in the data.

2) *Randomly Generated Data:* Similar results can be observed on the randomly generated data sets, where model selection using the attainment score performs best on 10 out of 16 data sets (see Table VI). Again, the performance drop on the remaining six data sets is predominantly due to an underestimation of the number of clusters. In the presence of cluster structures on several levels, MOCK tends to "pick" the level at which individual clusters are perfectly separated. This effect can be reduced by considering all local optima (not only the global optimum) in the attainment plot. This strategy was pursued in [14], resulting in a higher selection performance across the entire range of data sets studied.

C. Running Time Analysis

In this section, we analyze the worst-case time complexity of the algorithms for the generation of a set of solutions with $k \in \{1, \dots, k_{\text{user}}\}$ for a given data set of size N and dimensionality D , and the subsequent model selection phase. Table VII shows the complexity of the two main components of the algorithms (the clustering phase and the model selection phase), as well as

TABLE VI

NUMBER OF CLUSTERS k AND QUALITY OF THE SOLUTION (IN TERMS OF THE ADJUSTED RAND INDEX, AND WITHIN THE CLUSTER RANGE OF $k \in \{1, \dots, 25\}$) DETERMINED BY (FROM LEFT TO RIGHT) MOCK'S ATTAINMENT SCORE METHOD, THE SILHOUETTE WIDTH (APPLIED TO THE OUTPUT OF MOCK, k -MEANS, AVERAGE LINK, SINGLE LINK, AND THE CLUSTER ENSEMBLE), AND THE GAP STATISTIC. VALUES FOR MOCK, k -MEANS AND THE CLUSTER ENSEMBLE ARE MEAN VALUES FROM 21×10 RUNS; AVERAGE LINK AND SINGLE LINK ARE DETERMINISTIC, SO MEANS ARE CALCULATED OVER THE TEN DIFFERENT PROBLEM INSTANCES ONLY. THE STATISTICALLY BEST PERFORMER(S) ($\alpha = 0.005$) ARE HIGHLIGHTED IN BOLD FACE. TO TEST IF DIFFERENCES WERE STATISTICALLY SIGNIFICANT, A PAIRED WILCOXON SIGNED-RANK TEST [66], WHICH TAKES INTO ACCOUNT THE DEPENDENCE ON PROBLEM INSTANCE, WAS APPLIED TO ALL PAIRS OF ALGORITHMS

Problem	MOCK (att)		MOCK (sil)		k -means (sil)		Av. link (sil)		Single link (sil)		Ensemble (sil)		k -means (gap)	
	k	Rand	k	Rand	k	Rand	k	Rand	k	Rand	k	Rand	k	Rand
2d-4c	3.77	0.9151	3.7	0.8755	3.69	0.8706	4.5	0.8908	4.4	0.5015	2.23	0.2625	1.4	0.1588
10d-4c	3.94	0.9797	3.6	0.9176	3.59	0.9037	4	0.6352	2	0.0002	5.37	0.3934	1	0
50d-4c	4.49	0.9349	5.53	0.8126	2.9	0.3603	2.7	0.3650	6.2	0.0894	2.85	0.1335	1	0
100d-4c	4.98	0.9072	5.74	0.7779	3.2	0.3521	3.2	0.3739	1.6	7.9e-05	2.79	0.1579	1	0
2d-10c	8.78	0.8874	9.65	0.8445	10.66	0.7788	15.2	0.6862	8	0.1472	4.5	0.0793	2.13	0.1207
10d-10c	10.11	0.9272	8.88	0.8939	9.02	0.9147	5.3	0.2693	2	8.9e-05	3.86	0.2751	1	0
50d-10c	10.35	0.9732	14.3	0.8252	9.17	0.3545	6.4	0.2573	2	-3.2e-07	5.06	0.1086	1	0
100d-10c	11.37	0.9482	15.32	0.8065	9.17	0.3797	7.1	0.2681	2	1.4e-05	3.94	0.0943	1	0
2d-20c	13.85	0.7969	17.31	0.8986	21.87	0.8180	16.3	0.8389	16.3	0.6619	1.24	-4.6e-06	1	0
10d-20c	19.33	0.9775	19.56	0.9864	19.19	0.9623	19.8	0.9913	18.2	0.7993	3.69	0.1113	1	0
50d-20c	19.20	0.8018	33.67	0.7385	22.61	0.3930	25.3	0.3359	2.4	0.0010	22.6	0.7013	1	0
100d-20c	20.60	0.8292	30.18	0.7597	23.32	0.4243	16.3	0.2038	2.8	0.0010	14.53	0.6664	1	0
2d-40c	12.37	0.3811	35.26	0.8140	30.63	0.6364	30.9	0.7412	27.7	0.3670	2.27	0.0292	1	0
10d-40c	33.27	0.8858	37.21	0.9591	38.61	0.9476	39	0.9790	36.5	0.6836	5.36	0.1231	1	0
50d-40c	17.1	0.232157	61.76	0.7133	45.44	0.3282	26.7	0.1028	2	0.0002	17.03	0.2995	1	0
100d-40c	19.07	0.2658	57.41	0.7074	43.77	0.3211	26.1	0.1182	2.1	0.0003	15.18	0.3006	1	0

TABLE VII

WORST-CASE TIME COMPLEXITY OF THE ALGORITHMS FOR THE GENERATION OF A SET OF SOLUTIONS WITH $k \in \{1, \dots, k_{\text{user}}\}$ FOR A GIVEN DATA SET OF SIZE N AND DIMENSIONALITY D , AND THE SUBSEQUENT MODEL SELECTION PHASE. IN ADDITION TO N , D , AND k_{user} , THE TIME COMPLEXITY OF MOCK DEPENDS ON THE NUMBER OF EVALUATIONS E , THE SIZE P OF SOLUTION AND CONTROL FRONTS (WHICH IS BOUNDED BY THE EXTERNAL POPULATION SIZE USED), AND THE NUMBER OF REFERENCE DISTRIBUTIONS R CONSIDERED. THE TIME COMPLEXITY OF BOTH k -MEANS (SIL) AND k -MEANS (GAP) GROWS WITH ADDITIONAL FACTORS: THE NUMBER OF RESTARTS S , THE NUMBER OF ITERATIONS REQUIRED UNTIL CONVERGENCE I (WHICH VARIES UNPREDICTABLY DEPENDING ON THE PROPERTIES OF THE DATA SET), AND, IN THE CASE OF k -MEANS (GAP) ONLY, THE NUMBER OF REFERENCE DISTRIBUTIONS R CONSIDERED. THE TIME COMPLEXITY QUOTED FOR THE ENSEMBLE METHOD IS FOR THE MERGING OF LABELINGS ONLY

	MOCK (att)	MOCK (sil)	k -means (sil)	Av. link (sil)	Single link (sil)	Ensemble (sil)	k -means (gap)
Clustering phase	$O(N^2 \log ND k_{\text{user}}^2 E)$	$O(N^2 \log ND k_{\text{user}}^2 E)$	$O(ND k_{\text{user}}^2 IS)$	$O(N^2 D)$	$O(N^2 D)$	$O(N^2 k_{\text{user}}^5)$	$O(ND k_{\text{user}}^2 IS)$
Model selection	$O(N^2 \log ND^3 k_{\text{user}}^2 P^2 ER)$	$O(N^2 DP)$	$O(N^2 D k_{\text{user}})$	$O(N^2 D k_{\text{user}})$	$O(N^2 D k_{\text{user}})$	$O(N^2 D k_{\text{user}})$	$O(ND^3 k_{\text{user}}^2 ISR)$
Overall complexity	$O(N^2 \log ND^3 k_{\text{user}}^2 P^2 ER)$	$O(N^2 \log ND k_{\text{user}}^2 PE)$	$O(N^2 D k_{\text{user}}^2 IS)$	$O(N^2 D k_{\text{user}})$	$O(N^2 D k_{\text{user}})$	$O(N^2 D k_{\text{user}}^5)$	$O(ND^3 k_{\text{user}}^2 ISR)$

their overall complexity. The derivation of the complexities for MOCK is briefly discussed next. The complexities quoted for the other algorithms have been derived from simple considerations about our experimental setup and the known complexities from the literature.

MOCK generates the entire set of solutions with $k \in \{1, \dots, k_{\text{user}}\}$ in a single run, and therefore needs to be run only once. Its complexity grows linearly with the number of evaluations E used, which is the product of the number of iterations and the IP size and has been kept constant in our experiments. The complexity of the individual evaluations, which consist of the decoding step and the computation of the two clustering objectives, is $O(ND)$. For small data sets, it is these evaluations that dominate the running time of the algorithm. However, MOCK's clustering phase also requires a number of one-off computations, which have a high computational complexity and take up an increasing proportion of computation time for large data sets. In particular, during initialization, MOCK requires the computation of the complete dissimilarity matrix: $O(N^2 D)$; the construction of the MST: $O(N^2)$; the repeated running of a restricted version of k -means

(with no restarts and at most ten iterations, see Section III-A7): $O(ND k_{\text{user}}^2)$; and the computation of the nearest-neighbor lists: $O(N^2 \log N)$.

MOCK's phase of model selection requires the computation of the null model and the generation of R control fronts. In this paper, a Poisson model in eigenspace has been used as the null model, which results in a high computational complexity $O(ND^3)$, but it is worth noting that less expensive options exist.⁷ MOCK's attainment score method requires the computation of all distances between solutions in the solution and the control fronts, an operation quadratic in the size of these fronts, P , which are bounded by the EP size.

Thus, MOCK's overall complexity is $O(N^2 \log ND^3 k_{\text{user}}^2 P^2 ER)$. A number of representative running times for MOCK and the single-objective clustering algorithms from our experiments (the combined running time for both clustering and model selection) are shown in Table VIII.

⁷For example, in [12], a Poisson model in the original data space was used for the generation of the control data, which is of complexity $O(ND)$. The difference this makes to the model selection accuracy is demonstrated in [14].

TABLE VIII

REPRESENTATIVE RUNNING TIMES (IN SECONDS) ON AN 3.0 GHz INTEL PENTIUM 4 COMPUTER RUNNING RedHat ENTERPRISE LINUX. THESE DATA SETS HAVE BEEN SELECTED TO PROVIDE A GOOD OVERVIEW OF THE DIFFERENT SIZES AND DIMENSIONALITIES CONSIDERED IN OUR EXPERIMENTAL STUDY. ALL PROGRAMS ARE WRITTEN IN C EXCEPT FOR THE CLUSTER ENSEMBLE, WHICH IS WRITTEN IN MATLAB (THEREFORE, RUNNING TIMES FOR THE CLUSTER ENSEMBLE CANNOT BE FAIRLY COMPARED WITH THE OTHER ALGORITHMS, AND ARE NOT INCLUDED). RECALL THAT THE NOTATION x d - y d IN THE NAME OF A DATA SET INDICATES THAT IT IS OF DIMENSIONALITY x AND CONTAINS y CLUSTERS

Data				Running time in seconds						
Data set	N	D	k	MOCK (att)	MOCK (sil)	k -means (sil)	Av. link (sil)	Single link (sil)	Ensemble (sil)	k -means (gap)
2d-4c-no1	1623	2	4	280	84	51	14	13	n.a.	157
2d-10c-no1	2525	2	10	565	149	82	50	50	n.a.	256
10d-4c-no1	958	10	4	215	47	20	3	3	n.a.	88
10d-10c-no1	3565	10	10	1194	252	105	88	88	n.a.	367
100d-4c-no1	1286	100	4	994	202	240	8	8	n.a.	728
100d-10c-no1	2892	100	10	2654	548	491	81	82	n.a.	1639

VI. DISCUSSION

MOCK has shown a very good performance in terms of generating high-quality solutions consistently across the full range of our data sets. There are two alternative propositions to explain the superior performance of MOCK compared with the single-objective clustering algorithms considered in our study: (i) MOCK benefits from the use of multiple objectives; or (ii) MOCK is more effective at optimizing the individual objectives. A number of different pieces of evidence support (i) and indicate that (ii) is not true.

Proposition (i) follows directly if the two clustering criteria chosen for use in MOCK are generally in conflict. For a given data set where the objectives are in conflict, either it is the case that only one objective can lead to good solutions, or it is only tradeoffs between the two that can lead to good solutions. In the first case, MOCK and only the *appropriate* single-objective clustering algorithm(s) can succeed. In the second case, only MOCK can succeed. Thus, MOCK would have an advantage overall.

It should be clear that MOCK's two objectives, overall deviation (rewarding compact clusters) and connectivity (rewarding local connectedness of clusters), can only be perfectly correlated for very easy data sets containing well-separated, compact clusters (if solutions with a fixed number of clusters are considered). For more complex data that contain overlap between clusters, elongated cluster shapes, or differing extents of clusters, the degree of conflict between these two criteria increases because all of these factors violate the assumptions of one or the other of the two criteria. Moreover, when solutions with different numbers of clusters are considered, the two objectives will always be conflicting, even on easy data sets. This is because the objectives have opposite natural biases with respect to the number of clusters. The conflicting nature of the two criteria can also be readily seen from Figs. 6 and 9.

The effect that this conflict has is also demonstrated by our experimental results, in particular through our comparison against single link and k -means, which optimize objectives closely related to connectivity and overall deviation, respectively. If the two objectives were not conflicting, single link and k -means would yield the same (or similar) solutions (which they do not on any of the data sets considered in the experiments, and usually at least one of the pair returns a very poor solution). Moreover, the data sets where *all* the single-objective algorithms can

be said to fail, and where MOCK does substantially better than all of them, are those where it is most obvious that the best solution has to be a tradeoff between compactness and connectivity, e.g., LongSquare (where both compact clusters and elongated clusters exist in the same data set). Thus, both theory and results consistently point to the fact that MOCK is the more consistent performer as a direct consequence of it optimizing two conflicting criteria.

Our results also show that proposition (ii) above is not the case: on data sets that are consistent with k -means' optimization criterion, the algorithm usually outperforms MOCK, slightly. The same is true for single link. This would not be the case if MOCK were simply the "better" optimizer. The fact that (ii) is not valid can also be seen when comparing overall deviation and connectivity values for the three algorithms. Fig. 6 indicates the high performance of single link in terms of connectivity, and the high performance of k -means in terms of overall deviation (they usually outperform MOCK slightly in this respect).

These considerations lead to the conclusion that multiple objectives are useful for obtaining a consistent performance with respect to cluster generation. However, this is only one aspect of the clustering process: given the large number of tradeoff solutions generated by a multiobjective clustering approach, an effective method of solution selection is also essential. Fortunately, the multiobjective approach also lends itself naturally to a form of model selection that we have found to be highly effective. The method estimates the degree of genuine structure captured by a given clustering solution by comparing its two objective values to those achieved when clustering random control data containing no cluster structure.

Limitations and Applicability: Despite its promising performance, MOCK naturally has some inherent limitations. Due to the heuristic nature of the algorithm, MOCK provides an approximation to the real (unknown) Pareto front only. Therefore, generation of the best clustering solution cannot be guaranteed, and results show some (albeit small) variation between individual runs. This limitation also concerns our method of solution selection; this considers the distance between solution and control fronts obtained by the algorithm, and variations in either front can therefore have an impact on the solutions selected. This is where, currently, the main variation in the algorithm occurs, and an increased robustness would be desirable. Moreover, although we have investigated using other objectives than the two used here (and not found any to be as effective), it remains

possible that other criteria may be needed to perform well on certain data sets. In addition, the approach is too computationally expensive for some applications with heavy time or memory constraints, e.g., real-time document clustering.

Finally, it should be noted that the desirability of multiobjective clustering depends on the targeted application domain. In applications such as vector quantization, where very stringent assumptions can be made on the type of desirable clusters (e.g., highly compact, i.e., spherical) and the clustering problem is defined formally in terms of internal criteria such as minimum variance, the use of multiobjective clustering is not justified. In contrast, we see the advantages of multiobjective clustering in the much broader area of exploratory data analysis where very little is known *a priori* about the shapes, properties and number of the clusters present, and appropriate validation of results is desirable.

VII. CONCLUSION

In this paper, we have described an evolutionary approach to multiobjective clustering. We have discussed the conceptual advantages of multiobjective clustering and demonstrated that these translate into a performance advantage in practice: the proposed evolutionary approach has been shown to outperform traditional single-objective clustering techniques and an ensemble method across a diverse range of benchmark data sets. The algorithm has the additional advantage of keeping the number of clusters dynamic, and providing an automatic estimate of the quality of individual solutions. This can be used to perform model selection, and, in comparison to established model selection approaches, yields very promising results.

The software for MOCK is available from <http://www.dbk.ch.umist.ac.uk/handl/mock/>.

ACKNOWLEDGMENT

MOCK is built from D. Corne's original PESA-II code. The authors are grateful to A. Strehl for making available the ClusterPack MATLAB toolbox. They thank D. Kell for help with the preparation of this manuscript and general advice. We are grateful to the anonymous reviewers whose comments have helped to improve many aspects of this manuscript.

REFERENCES

- [1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
- [2] B. S. Everitt, *Cluster Analysis*. London, U.K.: Edward Arnold, 1993.
- [3] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
- [4] P. Arabie, L. J. Hubert, and G. D. Soete, *Clustering and Classification*. Singapore: World Scientific, 1996.
- [5] G. W. Milligan, "Clustering validation: Results and implications for applied analyses," in *Clustering and Classification*. Singapore: World Scientific, 1996, ch. 10, pp. 341–367.
- [6] A. Ferligoj and V. Batagelj, "Direct multicriterion clustering," *J. Classification*, vol. 9, pp. 43–61, 1992.
- [7] J. Handl and J. Knowles, "Multiobjective clustering around medoids," in *Proc. IEEE Congr. Evol. Comput.*, 2005, vol. 1, pp. 632–639.
- [8] J. Handl, J. Knowles, and D. B. Kell, "Computational cluster validation in post-genomic data analysis," *Bioinformatics*, vol. 21, no. 15, pp. 3201–3212, 2005.
- [9] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Machine Learn. Res.*, vol. 3, pp. 583–617, 2002.
- [10] A. Topchy, A. K. Jain, and W. Punch, "A mixture model for clustering ensembles," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 379–390.
- [11] J. Handl and J. Knowles, "Evolutionary multiobjective clustering," in *Proc. 8th Int. Conf. Parallel Problem Solving From Nature*, 2004, pp. 1081–1091.
- [12] —, "Multiobjective clustering with automatic determination of the number of clusters," Inst. Sci. Technol., Univ. Manchester, Manchester, U.K., Tech. Rep. TR-COMPSYSBIO-2004-02, 2004. [Online]. Available: <http://www.dbkweb.ch.umist.ac.uk/handl/publications.html>
- [13] —, "Exploiting the tradeoff—The benefits of multiple objectives in data clustering," in *Proc. 3rd Int. Conf. Evol. Multicriterion Optim.*, 2005, pp. 547–560.
- [14] —, "Improvements to the scalability of multiobjective clustering," in *Proc. IEEE Congr. Evol. Comput.*, 2005, vol. 3, pp. 2372–2379.
- [15] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surveys*, vol. 31, no. 3, pp. 264–323, 1999.
- [16] L. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.
- [17] E. Vorhees, "The effectiveness and efficiency of agglomerative hierarchical clustering in document retrieval," Ph.D. dissertation, Dept. Comput. Sci., Cornell Univ., Ithaca, NY, 1985.
- [18] T. Kohonen, *Self-Organizing Maps*, ser. Springer Series in Information Sciences. Berlin, Germany: Springer-Verlag, 2001, vol. 30.
- [19] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statist. Soc., ser. B*, vol. 39, no. 1, pp. 1–38, 1977.
- [20] G. McLachlan and T. Krishnan, "The EM algorithm and extensions," in *Wiley Series in Probability and Statistics*. New York: Wiley, 1997.
- [21] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. Int. Conf. Manage. Data*, 1999, pp. 49–60.
- [22] M. Ester, H.-P. Kriegel, and J. Sander, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, 1996, pp. 226–231.
- [23] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *J. Intell. Inf. Syst.*, vol. 17, no. 2, pp. 107–145, 2001.
- [24] E. Falkenauer, *Genetic Algorithms and Grouping Problems*. New York: Wiley, 1998.
- [25] L. O. Hall, I. B. Özyurt, and J. C. Bezdek, "Clustering with a genetically optimized approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 103–112, 1999.
- [26] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data*. New York: Wiley, 1990.
- [27] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognit.*, vol. 33, no. 9, pp. 1455–1465, 2000.
- [28] H. Pan, J. Zhu, and D. Han, "Genetic algorithms applied to multi-class clustering for gene expression data," *Genomics, Proteomics, Bioinformatics*, vol. 1, no. 4, pp. 279–287, 2003.
- [29] G. Fleurya, A. Hero, S. Zarepari, and A. Swaroop, "Gene discovery using Pareto depth sampling distributions," *J. Franklin Instit. (Special Number on Genomics, Signal Processing and Statistics)*, vol. 341, no. 1–2, pp. 55–75, 2004.
- [30] Y. Kim, W. N. Street, and F. Menczer, "Evolutionary model selection in unsupervised learning," *Intell. Data Anal.*, vol. 6, no. 6, pp. 531–556, 2002.
- [31] M. Morita, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, 2003, pp. 666–671.
- [32] V. Estivill-Castro, "Why so many clustering algorithms: A position paper," *ACM SIGKDD Explorations Newsletter Archive*, vol. 4, no. 1, pp. 65–75, 2002.
- [33] J. Kleinberg, "An impossibility theorem for clustering," in *Proc. 15th Conf. Neural Inf. Process. Syst.*, 2002. [Online]. Available: <http://www.books.nips.cc/>
- [34] M. H. Law, "Multiobjective data clustering," in *Proc. IEEE Comput. Soc. Conf. Pattern Recognit.*, 2004, pp. 424–430.
- [35] B. Minaei-Bidgolli, A. Topchy, and W. F. Punch, "Ensembles of partitions via data resampling," in *Proc. Int. Conf. Inf. Technol.: Coding Comput.*, 2004, pp. 188–192.
- [36] A. Topchy, A. K. Jain, and W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, Dec. 2005.
- [37] A. Topchy, B. Minaei, A. K. Jain, and W. Punch, "Adaptive clustering ensembles," in *Proc. Int. Conf. Pattern Recognit.*, 2004, pp. 272–275.

- [38] M. B. Dale and P. T. Dale, "Classification with multiple dissimilarity matrices," *Coenoses*, vol. 9, pp. 1–13, 1992.
- [39] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proc. Genetic Evol. Comput. Conf.*, 2001, pp. 283–290.
- [40] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Norwell, MA: Kluwer, 2002.
- [41] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, U.K.: Wiley, 2001.
- [42] X. Gandibleux, M. Sevaux, K. Soerensen, and V. T'Kindt, Eds., *Metaheuristics for Multiobjective Optimization*, ser. Lecture Notes in Economics and Mathematical Systems. Berlin, Germany: Springer-Verlag, 2004, vol. 535.
- [43] C. A. Coello Coello, List of references on evolutionary multiobjective optimization. [Online]. Available: <http://www.delta.cs.cinvestav.mx/~ccoello/EMOO/EMOObib.html>
- [44] M. Laumanns, E. Zitzler, and L. Thiele, "On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization," in *Proc. 1st Int. Conf. Evol. Multi-Criterion Optim.*, 2001, pp. 181–196.
- [45] E. Hughes, "Evolutionary many-objective optimization: Many once or one many?," in *Proc. IEEE Congr. Evol. Comput.*, 2005, vol. 1, pp. 222–227.
- [46] C. Ding and X. He, "K-nearest-neighbor consistency in data clustering: Incorporating local information into global optimization," in *Proc. ACM Symp. Appl. Comput.*, 2004, pp. 584–589.
- [47] Y.-J. Park and M.-S. Song, "A genetic algorithm for clustering problems," in *Proc. 3rd Annu. Conf. Genetic Program.*, 1998, pp. 568–575.
- [48] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.
- [49] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 2–9.
- [50] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, pp. 65–85, 1994.
- [51] R. J. Wilson and J. J. Watkins, *Graphs: An Introductory Approach: A First Course in Discrete Mathematics*. New York: Wiley, 1990.
- [52] J. Branke, K. Deb, H. Dierolf, and M. Osswald, "Finding knees in multi-objective optimization," in *Proc. 8th Int. Conf. Parallel Problem Solving From Nature*, 2004, pp. 722–731.
- [53] I. Das, "On characterizing the 'knee' of the Pareto curve based on normal-boundary intersection," *Structural Optim.*, vol. 18, no. 2–3, pp. 107–115, 1999.
- [54] K. Deb, "Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions," in *Advances in Evolutionary Computing: Theory and Applications*. London, U.K.: Springer-Verlag, 2003, pp. 263–292.
- [55] C. A. Mattson, A. A. Mullur, and A. Messac, "Smart Pareto filter: Obtaining a minimal representation of multiobjective design space," *Eng. Optim.*, vol. 36, no. 6, pp. 721–740, 2004.
- [56] G. Stehr, H. Graeb, and K. Antreich, "Performance trade-off analysis of analog circuits by normal-boundary intersection," in *Proc. 40th Design Autom. Conf.*, 2003, pp. 958–963.
- [57] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a dataset via the Gap statistic," *J. Royal Statist. Soc.: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.
- [58] W. S. Sarle, Cubic clustering criterion SAS Instit. Inc., Cary, NC, SAS Tech. Rep. A-108, 1983.
- [59] C. M. Fonseca and P. J. Fleming, "On the performance assessment and comparison of stochastic multiobjective optimizers," in *Proc. 4th Int. Conf. Parallel Problem Solving From Nature*, 1996, pp. 584–593.
- [60] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, pp. 53–65, 1987.
- [61] J. M. Pena, J. A. Lozana, and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recognit. Lett.*, vol. 20, no. 10, pp. 1027–1040, 1999.
- [62] A. Strehl, ClusterPack Matlab Toolbox. [Online]. Available: <http://www.lans.ece.utexas.edu/~strehl/soft.html>
- [63] J. Ghosh, A. Strehl, and S. Merugu, "A consensus framework for integrating distributed clusterings under limited knowledge sharing," in *Proc. NSF Workshop Next Generation Data Mining*, 2002, pp. 99–108.
- [64] W. Rand, "Objective criteria for the evaluation of clustering methods," *J. Amer. Statist. Assoc.*, vol. 66, no. 336, pp. 846–850, 1971.
- [65] A. Hubert, "Comparing partitions," *J. Classification*, vol. 2, pp. 193–198, 1985.
- [66] W. J. Conover, *Practical Nonparametric Statistics*, 2nd ed. New York: Wiley, 1980.



Julia Handl received the B.Sc. (Hons.) degree in computer science from Monash University, Victoria, Australia, in 2001 and the M.Sc. degree in computer science from the University of Erlangen-Nuremberg, Germany, in 2003. She is currently working towards the Ph.D. degree at the University of Manchester, Manchester, U.K., where her work is focused on multiobjective approaches to unsupervised and semisupervised classification.



Joshua Knowles received the B.Sc. (Hons.) degree in physics with mathematics, the M.Sc. degree in information systems engineering (at Distinction level), and the Ph.D. degree from the University of Reading, Reading, U.K., in 1993, 1996, and 2002, respectively.

From 2001 to 2003, he was a Marie Curie Fellow at IRIDIA, Free University of Brussels, and since 2003, he has held a BBSRC David Phillips Fellowship at the University of Manchester, Manchester, U.K. He is currently serving on the U.K. EPSRC Peer Review College. His main research interest

is heuristic, multiobjective optimization and its application to computational biology.