An improved and more scalable evolutionary approach to multiobjective clustering

Garza-Fabre, Mario; Handl, Julia; Knowles, Joshua

DOI:
10.1109/TEVC.2017.2726341

An Improved and More Scalable Evolutionary Approach to Multiobjective Clustering

Mario Garza-Fabre, Julia Handl, and Joshua Knowles

***Abstract*—The multiobjective realisation of the data clustering problem has shown great promise in recent years, yielding clear conceptual advantages over the more conventional, single-objective approach. Evolutionary algorithms have largely contributed to the development of this increasingly active research area on multiobjective clustering. Nevertheless, the unprecedented volumes of data seen widely today pose significant challenges and highlight the need for more effective and scalable tools for exploratory data analysis. This paper proposes an improved version of the *multiobjective clustering with automatic k-determination* algorithm. Our new algorithm improves its predecessor in several respects, but the key changes are related to the use of an efficient, specialised initialisation routine and two alternative reduced-length representations. These design components exploit information from the minimum spanning tree and redefine the problem in terms of the most relevant subset of its edges. Our study reveals that both the new initialisation routine and the new solution representations not only contribute to decrease the computational overhead, but also entail a significant reduction of the search space, enhancing therefore the convergence capabilities and overall effectiveness of the method. These results suggest that the new algorithm proposed here will offer significant advantages in the realm of 'big data' analytics and applications.**

***Index Terms*—Evolutionary Computation, Data Analysis, Clustering Methods, Data Mining, Pareto Optimization.**

I. Introduction

DATA, whether generated *e.g.* by customer transactions, through communications on social media, or as a by-product of manufacturing processes, are generally meaningless unless suitable techniques are employed to select, analyse, and transform these raw data into tangible information and insight. Data clustering is one of the most fundamental tools in exploratory data analysis, being concerned with finding homogeneous groups of data entities according to '*measured or perceived intrinsic characteristics*' [1]. Clustering has found important applications in many different areas such as marketing [2], [3], bioinformatics [4]–[6], and medicine [7], [8].

Crisp clustering (by far the main type of cluster analysis) requires that each datum be assigned to exactly one cluster, thus creating a partitioning of the input data [9], [10]. It is known that this crisp clustering is ill-posed in some strict sense [11]. More practically, there is often a trade-off between inter-cluster separation and intra-cluster homogeneity, as well as trade-offs related to the choice of the number of clusters to partition into, and the ability to optimise the other two criteria. It was against this background that Handl and Knowles developed a multiobjective approach to clustering [12]–[15], although some earlier independent work existed [16].

In this paper, we revisit the *multiobjective clustering with automatic k-determination* (MOCK) algorithm [15]. The original version of MOCK was described in [14], with further adjustments to its initialisation, mutation, and solution selection schemes formalised in [15], [17]. In its standard form, MOCK is limited to the application to numerical data, but a more flexible version, MOCK around medoids, was described in [18]. Here, we suggest modifications to the version of MOCK introduced in [15] with a view to improving its scalability, which is essential given the unprecedented volumes of data that require processing in current clustering applications. In particular, the methodology described in this paper is based on a thorough analysis of the core mechanisms of MOCK, specifically the interplay of its representation, initialisation routine, optimiser and evaluation functions, and the associated identification of possible efficiencies in the process. Our work can be seen as complementary to adjustments such as those introduced in [19], which achieve improved scalability by using a sub-sample of the data to guide the clustering process.

Our new algorithm version, Δ-MOCK, presents extensive changes, including changes to the multiobjective optimisation engine and evaluation criteria, but, more fundamentally, to the initialisation and representation schemes; despite these significant adaptations, the core principles of Δ-MOCK remain the same as in MOCK. This study conducts a comprehensive investigation and comparison of Δ-MOCK with respect to MOCK. Our findings demonstrate that the changes introduced impact positively not only on computational efficiency but also on clustering accuracy for complex data sets.

A. Scope of this Study

The ultimate goal of this study is to investigate and illustrate the advantages of Δ-MOCK. This is achieved by evaluating this method, specifically, with respect to MOCK [15]. MOCK is not only a mandatory baseline (being the starting point for the development of Δ-MOCK), but is also representative of the state-of-the-art. In addition, two clustering algorithms from the specialised literature have been included in our comparative analysis as a reference; this includes the well-known k-means method [20] and a multiobjective genetic algorithm for clustering [21]. Our analysis focuses only on the ability (and efficiency) of these methods to produce candidate solution sets which contain high-quality partitions. Hence, our study centres on the so-called clustering (or model generation) phase of MOCK and Δ-MOCK, and is not concerned with the subsequent model selection phase of these methods (see

M. Garza-Fabre and J. Handl are with the Decision and Cognitive Sciences Research Centre, University of Manchester, Manchester, M15 6PB, UK. E-mail: mario.garza-fabre@manchester.ac.uk, julia.handl@manchester.ac.uk.

J. Knowles is with the School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK. E-mail: j.knowles@cs.bham.ac.uk.

Manuscript received MM DD, AAAA; revised MM DD, AAAA.

Section II-A). Finally, the versions of MOCK and Δ-MOCK studied here assume that input data entities are represented by vectors of numerical attributes; adaptations (such as those studied in [18]) to scenarios where data can be described by non-numerical attributes, or where only dissimilarity (or similarity) data describing the relationships between entities is available, are considered beyond the scope of this study.

B. Organisation of this Paper

The remainder of this paper is organised as follows. First, Section II introduces the necessary background, sets the notation, and discusses relevant related works. Section III provides a detailed description of our Δ-MOCK algorithm, contrasting its differences with respect to MOCK. In Section IV, we describe the specific configuration and settings of the approaches evaluated and compared in this study, the benchmarks considered, and the performance assessment methodology adopted. The results of our experimental evaluation are presented in Section V. Then, the main findings of this evaluation are further discussed in Section VI along with some open questions that are expected to trigger future research efforts. Finally, Section VII concludes.

II. Background Concepts and Related Work

This section presents a more formal definition of the problem addressed in this paper. Also, this section introduces some basic concepts which are essential for the sake of self-containedness and provides a discussion of relevant literature.

A. Single-Objective and Multiobjective Clustering

Data clustering is the unsupervised task concerned with classifying a collection of data entities (or points), based on some notion of similarity, into a finite number of disjoint subsets called *clusters*.[1] When posed as an optimisation problem, this task usually relies on the optimisation of a single (*internal*[2]) clustering criterion, commonly referred to as *validity index* [22], which serves as a *proxy* for the unknown correct classification of the data. Formally, this task can be stated as the problem of determining C^* such that:

$$C^* \quad = \quad \arg\min_{C \in \Omega} f(C) \ , \tag{1}$$

where $C \in \Omega$ is a clustering solution; more specifically, C is a partition of the collection of N input data entities into k clusters which represent our approximation to the set of k^* true natural clusters (or classes) in the data. Ω is the space of all feasible clusterings (in this case, all proper, non-fuzzy partitions of the data), and $f : \Omega \rightarrow \mathbb{R}$ is a clustering criterion which, without loss of generality, is to be minimised.

In practice, however, a single criterion is generally unable to capture all desirable aspects of a clustering solution. This fact has motivated the consideration of alternative multiobjective formulations of the problem, relying on the simultaneous optimisation of multiple clustering criteria [15], [16], [23]. With such formulations, thus, we aim to find C^* such that:

$$C^* \quad = \quad \arg\min_{C \in \Omega} \mathbf{f}(C) \ , \tag{2}$$

where $\mathbf{f}(C) = [f_1(C), \ldots, f_m(C)]^T$ is a vector function and $f_i : \Omega \rightarrow \mathbb{R}$ is the i-th clustering optimisation criterion, $i \in \{1, \ldots, m\}$. Rather than searching for a single optimal partition, in the multiobjective context the task now becomes that of identifying a set of partitions yielding different trade-offs between the (usually conflicting) clustering criteria. More formally, the goal is to find a set of *Pareto-optimal solutions* $\mathcal{P}^*$, such that $\mathcal{P}^* = \{C^* \in \Omega \mid \nexists C \in \Omega : C \prec C^*\}$. The symbol '$\prec$' denotes the *Pareto-dominance* relation [24]:

$$\begin{aligned} C \prec C' \quad \Leftrightarrow \quad & \forall i : f_i(C) \leq f_i(C') \ \ \wedge \\ & \exists j : f_j(C) < f_j(C') \ , \\ & i, j \in \{1, \ldots, m\} \ . \end{aligned} \tag{3}$$

If $C \prec C'$, then C is said to *dominate* C'. Otherwise, C' is said to be *nondominated* with respect to C ($C \not\prec C'$). The image of $\mathcal{P}^*$ in objective space is the so-called *Pareto-optimal front*.

Note that $\mathcal{P}^*$ always comprises the optimal solutions for the m individual optimisation criteria. Therefore, a clustering method relying on a multiobjective formulation has the potential to produce the same optimal partitions that can be reached through the independent (single-objective) optimisation of these criteria. More interestingly, though, the multiobjective formulation enables the discovery of trade-offs which are otherwise difficult to obtain using single-objective strategies.

Let $\mathcal{P}$ be the solution set obtained by a given multiobjective clustering method. Since we cannot ensure that $\mathcal{P} = \mathcal{P}^*$, we say that $\mathcal{P}$ is an *approximation set* and its image in objective space is a *Pareto front approximation* (PFA). We will refer to the process of generating this set of candidate partitions as the *clustering (model generation) phase* of the clustering method. The process of choosing one or a subset of the most promising candidates from this set (the ultimate goal in many practical applications) will be referred to as the *model selection phase*.

B. Evolutionary Multiobjective Clustering

Owing to their well-known flexibility and effectiveness at solving difficult optimisation problems, evolutionary algorithms and other nature-inspired metaheuristics have been a popular choice for the design of data clustering techniques [25]–[28]. The suitability of these approaches is even more evident in the specific context of multiobjective clustering, as they are able to generate the target PFA within a single execution. Early works have clearly illustrated the conceptual advantages of *evolutionary multiobjective clustering* (EMC) [14], [15], [21], and have motivated intensive research in this area during the last decade [25], [29], [30].

In particular, the MOCK algorithm [15] exploits the benefits of EMC to a large extent. By optimising clustering criteria which present conflicting biases with respect to the value of k (see details in Section III-C), a single run of this method produces a PFA which not only comprises interesting trade-offs between the criteria optimised, but also can include

[1] This task is more specifically known as *crisp clustering*, in contrast to *fuzzy clustering* in which a single data point may belong to multiple clusters.

[2] Clustering algorithms, being unsupervised methods, do not use any kind of *external knowledge* about the correct (real) partition of the data.

partitions with a wide range of numbers of clusters. This removes the need for predefining this parameter, which is especially relevant given that we generally have little (or poor) information about how to choose k so it corresponds to k^*.

MOCK has been shown to produce solutions which significantly improve upon the quality of those obtained by single-objective and ensemble clustering approaches [15]. Despite the promising results that MOCK has shown in practice, we have identified concrete opportunities for increasing its performance and efficiency. Such opportunities have led to the development of the Δ-MOCK algorithm proposed in this paper.

We refer the reader to [15] for a comprehensive, self-contained description of MOCK. In Section III, we provide a brief description of this algorithm, with a particular emphasis on those aspects of MOCK which have been improved, highlighting the motivation and contrasting the differences with respect to the new design choices in our Δ-MOCK algorithm.

C. Representations for Evolutionary Clustering

A variety of solution representations for data clustering have been proposed in the literature [25], [30], with a comparison of the heuristic of four of them provided in [31]. Possible approaches range from the direct encoding of cluster memberships to the specialised encodings used in Falkenauer's grouping genetic algorithm [32]. Challenges of a direct encoding include the design of appropriate operators and the fact that these representations are *non-synonymously redundant* (see [33] for a definition and discussion of non-synonymous redundancy), problems that are overcome only partially in Falkenauer's approach. Moreover, both types of encodings require *a priori* decisions on the expected number of clusters.

An alternative, and increasingly popular, encoding has been based on the use of (continuous) cluster centroids, an approach that is thought to provide better scalability to large data sets: in particular, the length of the encoding depends on the dimension of the data and the number of clusters, but not on N. On the other hand, limitations of this approach are the assumption of spherical clusters that is implicit to a centroid-based approach, as well as the continuing requirement to specify the desired number of clusters.

MOCK [15] uses the *locus-based adjacency representation* originally proposed by Park and Song [35]. This encoding is illustrated in Fig. 1. Strengths of this approach include the straightforward definition of meaningful genetic operators, the capacity to capture arbitrary cluster shapes, and the ability to *naturally* encode partitions of varying k. This latter characteristic is of utmost importance in EMC, and particularly in the context of MOCK as discussed in Section II-B. Moreover, this representation has been found to be less redundant than other encodings and has been observed to contribute to the *heritability* and *locality* [36] of MOCK's operators [31].

Nevertheless, the locus-based adjacency representation has also been criticised for the fact that genome length increases linearly with the problem size N [29], [30]. This paper makes concrete why, due to specific strategies adopted during initialisation and genetic variation (see discussion in Sections III-G and V-E), this linear growth in the encoding (and

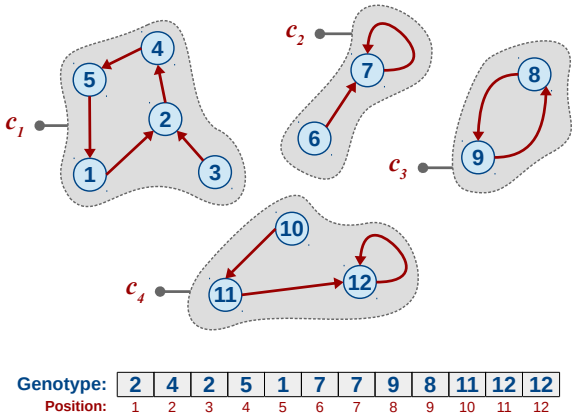

Fig. 1. Locus-based adjacency representation scheme used by MOCK. In this example, a data set consisting of $N = 12$ data points is considered. Data points are seen as the nodes of a graph. The genotype of an individual $\mathbf{x}$ consists of N genes, $x_1, \ldots, x_N$, where $x_i = j$ denotes a link $i \rightarrow j$ from data point i to data point j, $j \in \{1, \ldots, N\}$. A partition is derived by identifying all connected components in the resulting graph (which can be performed in linear time [34]), where each connected component is interpreted as a different cluster. Thus, the genotype of this example encodes a partition with $k = 4$ clusters. The graph in this figure is shown as a directed graph to aid in understanding how it originates from its corresponding genotype.

the resulting size of the search space) has not represented a major bottleneck to scalability (as is evident from MOCK's existing performance on large data sets with $N \approx 4{,}000$ [15]). Our recent preliminary work has revealed, however, that a reduced-length genetic representation can further improve the scalability of MOCK, providing clear benefits in terms of both clustering performance and computational efficiency [37].

The reduced-length encoding studied in [37] is revisited and further evaluated in this paper. Such an approach is here referred to as the Δ-locus encoding, and is one of the two alternative reduced-length representations implemented by our new algorithm Δ-MOCK, see Section III-B. The second approach, explored for the first time in this paper, is called the Δ-binary encoding. It has been found that a full-length version of our Δ-binary encoding is equivalent to the representation used by the independent work of Casillas *et al.* [38]. Such a representation proposed in [38] is in turn inspired by the early work of Caliński and Harabasz [39], where the space of possible clustering solutions is reduced to the set of partitions which can be obtained by splitting the *minimum spanning tree* (MST). To the best of our knowledge, these are the works more closely related to the representations studied in this paper.

D. Use of Concepts from Graph Partitioning in Clustering

There is a close relationship between the problems of data clustering and graph partitioning. In particular, graphs may be used to directly represent a data set: a given data set with N elements is described as a fully connected weighted graph with each of the N nodes representing a data element and the edge weights representing all pairwise dissimilarities. A partitioning of this graph may then directly be interpreted as a clustering of the data set (as illustrated in Fig. 1). This is best exemplified *e.g.* through the single-link hierarchical clustering algorithm: for a given number of clusters k, the results of this method

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2726341, IEEE Transactions on Evolutionary Computation

are equivalent to those from removing the $k-1$ longest links in an MST of the associated graph [40]. Alternative partitions may be obtained by adjusting the criterion that is guiding the successive removal of links from the MST [41]–[43].

As stated by Zahn, we can therefore address the '*problem of deleting edges from an MST so that the resulting connected subtrees correspond to the observable clusters*' [44]. This specific idea has been followed in [15], where a measure of *interestingness* and partitions initially obtained with k-means have been the criteria adopted to guide the removal of MST links during the generation of MOCK's initial solution set. Our Δ-MOCK algorithm explores this further: we propose an improved criterion for the identification of relevant MST links, and we exploit this notion not only to guide initialisation but also within the design of Δ-MOCK's representations.

III. Δ-MOCK

This section introduces Δ-MOCK and contrasts its differences with respect to its predecessor MOCK reported in [15].

The (clustering phase of) Δ-MOCK is outlined in Algorithm 1.[3] The algorithm starts by loading the input data set and by precomputing information and data structures which are exploited during subsequent stages (lines 1 and 2 in Algorithm 1). This involves the dissimilarity matrix, the ranked lists of nearest neighbours for all data points, and the MST upon which Δ-MOCK so heavily relies (Prim's algorithm is used for the MST computation). Then, the search strategy of Δ-MOCK is applied (lines 3 to 9 in Algorithm 1) in order to generate a PFA. This search strategy and all other design components of Δ-MOCK are separately described below.

Algorithm 1 Δ-MOCK (clustering phase).

1: $data\_loading()$
2: $initial\_precomputations()$
3: $\mathcal{P} \leftarrow initialisation(P)^{\dagger}$
4: **for** $generation \leftarrow 1$ **to** G_{max} **do**
5: $\quad \hat{\mathcal{P}} \leftarrow mating\_selection(\mathcal{P})$
6: $\quad \mathcal{P}' \leftarrow genetic\_operators(\hat{\mathcal{P}})$
7: $\quad \mathcal{P} \leftarrow survival\_selection(\mathcal{P} \cup \mathcal{P}')$
8: **end for**
9: $\mathcal{P} \leftarrow Pareto\_nondominated(\mathcal{P})$ $\qquad {}^{\dagger}|\mathcal{P}| = P$

A. Optimisation Engine

The original MOCK [15] is based on the *Pareto envelope-based selection algorithm version 2* (PESA-II), a representative approach from the evolutionary multiobjective optimisation (EMO) literature [45]. However, PESA-II is strongly elitist: it is observed that the high selection pressure induced by this method prevents a fraction of the genetic material introduced during initialisation from being exploited and disseminated throughout the evolutionary process.[4] As detailed in Section III-E, Δ-MOCK uses a specialised initialisation routine which generates high-quality base partitions. In such a scenario, it becomes essential to capitalise on all of the genetic material that these highly optimised initial solutions may comprise. Accordingly, and although the choice of search algorithm should not be a matter of concern in the absence of these special optimisation conditions, Δ-MOCK adopts a different method as its underlying search engine: the *nondominated sorting genetic algorithm version 2* (NSGA-II) [46]. It is worth noting that the core ideas of MOCK/Δ-MOCK are not linked to a specific search algorithm. In principle, therefore, a variety of existing algorithms from the EMO literature can be used. Both PESA-II and NSGA-II have proven to be highly effective optimisers, particularly in scenarios like this where a reduced number of optimisation criteria are considered.

In Δ-MOCK's search strategy based on NSGA-II, all the genetic material obtained through our specialised initialisation routine forms the basis of the initial parent population. Once the initial parent population $\mathcal{P}$ is constructed, $|\mathcal{P}| = P$, the evolutionary process consists of a total of G_{max} generations. Broadly, at each generation an offspring population $\mathcal{P}'$ is created by mating selection and the genetic operators. Then, offspring compete against parent individuals in order to survive from one generation to the next. In this study, binary tournament selection is used as the mating selection strategy. The specifics of Δ-MOCK's initialisation and genetic operators are respectively discussed in Sections III-E and III-F.

The distinctive characteristic of NSGA-II is the use of the *nondominated sorting* procedure to drive selection, as well as the *crowding distance* (a secondary selection criterion) to promote the diversity and distribution of the solutions along the PFA obtained. This is exploited in Δ-MOCK with the aim of producing a range of different trade-offs between the clustering criteria considered, as well as a set of candidate partitions presenting potentially different numbers of clusters.

B. Genetic Representations

The length of the genotype in the locus-based adjacency representation (Fig. 1), given by the size of the problem, N, pinpoints one of the main limiting factors regarding the scalability of MOCK. Not only can the encoding length impact on the computational efficiency during the processing of candidate individuals, but, more importantly, the search space grows very rapidly as a function of N. This can affect convergence and might therefore compromise the ability of MOCK to reliably solve large problem instances. Although MOCK's initialisation routine and genetic operators adopt specific strategies which, implicitly, help overcome these difficulties (see discussion in Section III-G), Δ-MOCK additionally implements reduced-length encodings in order to, explicitly, cope with these issues.

[3] Δ-MOCK uses the same model selection strategy as MOCK. This study focuses only on the clustering phase as stated in Section I-A. Refer to [15] for a comprehensive description and evaluation of the model selection phase.

[4] PESA-II [45] uses an elitist external population (EP) to store the current PFA. An internal population (IP) is constructed anew each generation based on EP and the genetic operators. In MOCK [15], all the partitions generated during initialisation are candidates to become the initial members of EP. Since EP admits only nondominated solutions, all dominated candidates are filtered and discarded (prematurely) at this early stage of the search process.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2726341, IEEE Transactions on Evolutionary Computation

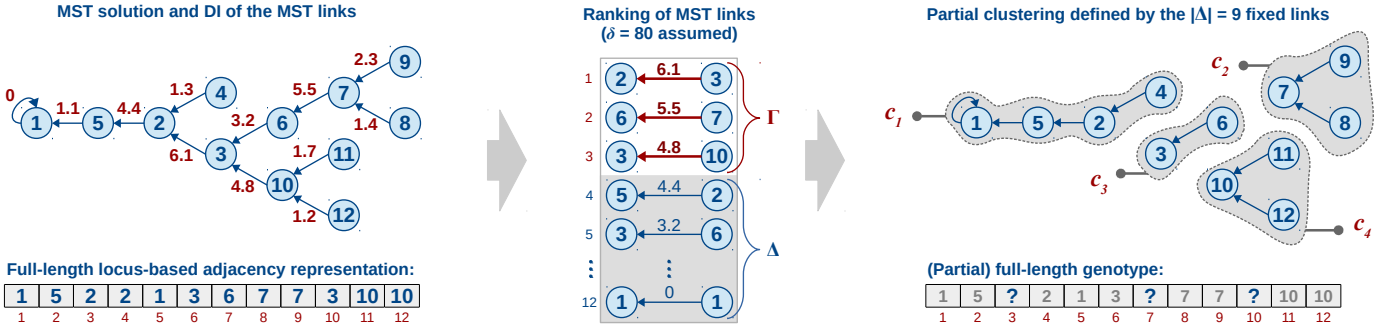

Fig. 2. Identification of the sets Γ and Δ of MST links. The MST is seen as a single-cluster (one connected component) solution, and the relevance of every MST link is computed on the basis of criterion DI. All the MST links are ranked in descending order of DI and then split into two disjoint sets based on parameter δ ($0 \leq \delta \leq 100$). The non-relevant (fixed) set Δ is formed by (approximately) the $\delta\%$ least interesting links, and the relevant set Γ is formed by the remaining, most interesting (top-ranked) links. In this example, a value of $\delta = 80$ is adopted, which produces a fixed set Δ and a relevant set Γ with nine and three MST links, respectively. The nine links in Δ, and their corresponding genes in the full-length genotype, are assumed fixed and lead to a partial clustering (with $k = 4$ clusters) which forms the basis for all candidate clustering solutions that Δ-MOCK explores during the evolutionary process.

Δ-MOCK introduces two alternative representations: the Δ-*locus encoding* and the Δ-*binary encoding*. Both the new encodings are based on the original representation of MOCK, but can significantly reduce the length of the genotype through the exploitation of information from the MST. More specifically, the Δ-locus and Δ-binary encodings rely on the identification of the most relevant subset of MST links whose removal, in an ideal scenario, can split the MST into a set of connected components that corresponds to the real cluster structure. The task of identifying such relevant links is not straightforward; different criteria have been used in the literature for these purposes as discussed in Section II-D, and success of a given criterion will certainly depend on the specific characteristics of the data. We propose here to use the *degree of interestingness* (DI) as a means to discriminate between the MST links; this criterion has been observed to provide better results than other criteria in this particular context (see Section V of our supplementary material). The concept of DI was previously explored in [15] with the aim of guiding part of the initialisation process of MOCK. Here, a more fine-grained version of this approach is considered. Formally, the DI of a link $i \rightarrow j$ is given by:

$$di(i \rightarrow j) \quad = \quad \min\left\{nn_i(j), nn_j(i)\right\} + \frac{\sigma(i,j)}{\sigma_{max}} \quad , \qquad (4)$$

where $nn_a(b)$ refers to the ranking of data point b in the list of nearest neighbours of data point a; in other words, $l = nn_a(b)$ indicates that b is the l-th nearest neighbour of a, also denoted as $b = nn_{al}$. By itself, the first term in (4) corresponds to DI as defined in [15]. The second term in this equation enables a better discrimination of MST links by explicitly considering the dissimilarity (distance) between points i and j, $\sigma(i,j)$. Note that dissimilarities are scaled to the range $[0,1]$ by dividing by the maximum dissimilarity found in the data set, $\sigma_{max} = \max\{\ \sigma(a,b)\ \mid\ a,b \in \{1,\ldots,N\}\ \}$.

Having defined DI, this criterion is used to rank and classify the MST links either into the set of relevant links, Γ, or into the set of non-relevant, fixed links, Δ. As exemplified in Fig. 2, such a classification depends not only on criterion DI, but on the setting of a user-defined parameter, namely δ. Parameter δ takes values from the range $[0,100]$ and plays a decisive role in Δ-MOCK. As can be seen from the figure, Γ comprises the $|\Gamma| = \lceil \frac{(100-\delta)}{100} N \rceil$ most prominent (highest DI) links, whereas Δ consists of the $|\Delta| = \lfloor \frac{\delta}{100} N \rfloor$ links with the lowest DI values. Furthermore, parameter δ determines the length of the genotype in the Δ-locus and Δ-binary encodings, which, as illustrated in Fig. 3, corresponds to the cardinality of the relevant set Γ. As explained in detail in the figure, the new encodings of Δ-MOCK redefine the problem in terms of the relevant MST links only, while all other non-relevant MST links (*i.e.* the set Δ) are considered to be a fixed, common characteristic of all candidate clustering solutions. By excluding all non-relevant MST links from the genotype, hence, only the removal (or replacement) of relevant links is optimised throughout the search process; as discussed before, this divides the MST into connected components which potentially translate into promising candidate partitions.

Therefore, the reduced-length encodings allow Δ-MOCK to concentrate its efforts on a (potentially small) relevant region of the solution space. Note that the Δ-locus encoding is equivalent to the locus-based adjacency representation when $\delta = 0$. Note also that the Δ-locus encoding has been previously explored in the context of MOCK [37], showing very encouraging results (although the approach studied in [37] used the original definition of DI provided in [15]). Finally, the full-length version ($\delta = 0$) of the Δ-binary encoding is similar to the representation used in [38], as discussed in Section II-C.

C. Optimisation Criteria

Aiming at the multiobjective essence of the problem of data clustering, MOCK relies on the optimisation of two complementary clustering criteria [15]: *overall deviation* (ODV) and *connectivity* (CNN). Whereas Δ-MOCK preserves the latter criterion as in MOCK, the former, ODV, is replaced with the *intra-cluster variance* (VAR) criterion. Although conceptually similar criteria, VAR is chosen to replace ODV in Δ-MOCK as it facilitates *delta-evaluation*. As discussed in Section III-D, this carries additional advantages in computational efficiency.

VAR and CNN evaluate fundamentally different but equally desirable properties of a clustering solution. Similarly to ODV,

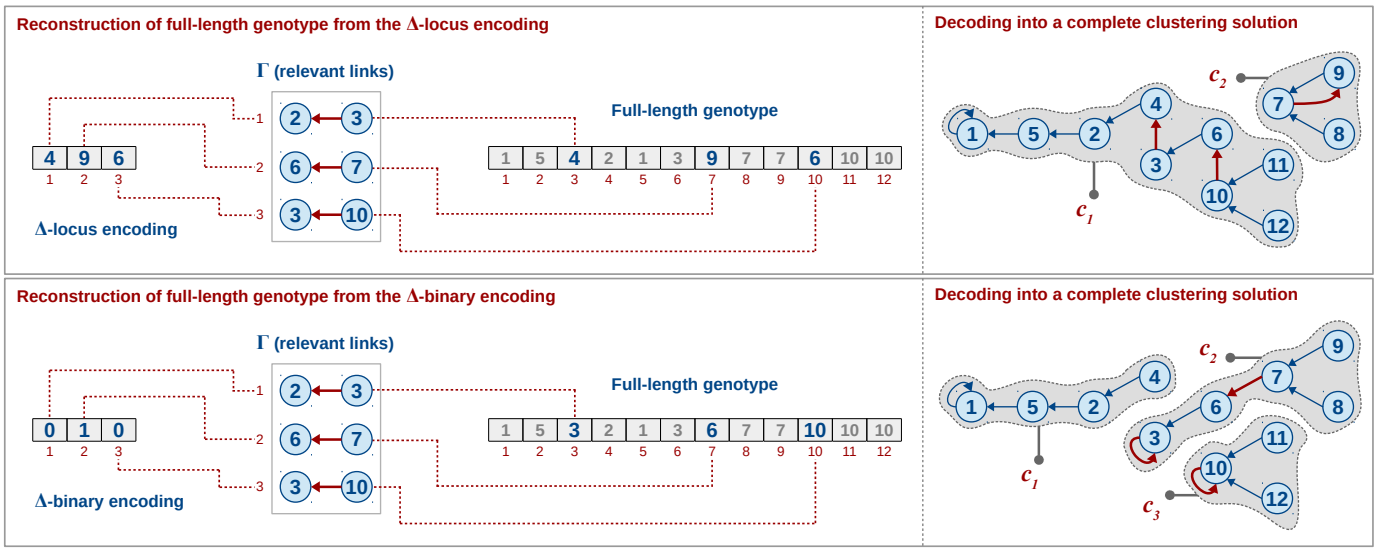

Fig. 3. The Δ-locus and Δ-binary reduced-length encodings, and the process of reconstructing the full-length genotype which is a first step towards decoding. The new encodings operate on genotypes of length $|\Gamma|$, where Γ is the set of relevant MST links as shown in Fig. 2. The partial clustering defined by the set of fixed MST links Δ plays an important role during the reconstruction of the full-length genotype. Using such a partial clustering as the starting point, the $|\Gamma|$-length genotypes of the new representations are used to define the missing pieces of information in the full-length genotype, which is then decoded into a complete clustering solution. In the Δ-locus encoding, genes $x^{\Delta}_1, \ldots, x^{\Delta}_{|\Gamma|}$ of an individual $\mathbf{x}^{\Delta}$ may assume any allele value from the set $\{1, \ldots, N\}$. In contrast, only allele values from the set $\{1, 0\}$ are considered when using the Δ-binary encoding. Thus, while the Δ-locus encoding is able to explore the replacement of the original MST links with other alternative links between data points, the Δ-binary encoding explores only whether the original MST links are preserved (allele value 1) or removed (allele value 0); removal of an MST link $i \rightarrow j$ is achieved by replacing this link with a self-connecting link $i \rightarrow i$.

VAR accounts for cluster compactness (also referred to as homogeneity of clusters), and is formally defined as follows:

$$var(C) \quad = \quad \frac{1}{N} \sum_{c \in C} v(c) \ , \tag{5}$$

where C is a candidate partition and $v(c)$ represents the individual contribution of a cluster $c \in C$ to this measure:

$$v(c) \quad = \quad \sum_{i \in c} \sigma(i, \mu_c)^2 \ . \tag{6}$$

In (6), μ_c denotes the centroid of cluster c, and $\sigma(i, \mu_c)$ refers to the dissimilarity between data point i and μ_c (the Euclidean distance is used as the dissimilarity measure in this study).

CNN captures cluster connectedness, reflecting the degree to which neighbouring points are identified as members of the same cluster. More formally, this measure is given by:

$$cnn(C) \quad = \quad \sum_{i=1}^{N} \sum_{l=1}^{L} \rho(i, l) \ , \tag{7}$$

where L is a user-defined parameter specifying the size of the neighbourhood to consider, and $\rho(i, l)$ penalises the clustering solution whenever data point i and its l-th nearest neighbour (denoted by nn_{il}) are not assigned to the same cluster:

$$\rho(i, l) \quad = \quad \begin{cases} \frac{1}{l}, & \text{if } \nexists c \in C \mid i \in c \wedge nn_{il} \in c \ ; \\ 0, & \text{otherwise} \ . \end{cases} \tag{8}$$

As objectives, both VAR and CNN are to be minimised. The independent optimisation of VAR tends to increase the number of clusters k. The best value for this measure can be achieved by making $k = N$, *i.e.* by isolating each datum in a different cluster. Oppositely, CNN presents the tendency of decreasing k, reaching its lowest value when all data points are clustered together, *i.e.* $k = 1$.[5] Therefore, the simultaneous optimisation of VAR and CNN in Δ-MOCK is expected to compensate for the individual biases of these criteria, producing a PFA of trade-off partitions with a diversity of values for k.

D. Delta-Evaluation

The reformulation of the clustering problem as a function of the relevant MST links only, makes it possible to precompute information regarding all other non-relevant MST links which are assumed invariant for all candidate solutions. Such non-relevant, fixed links, as illustrated in Fig. 2, define a partial clustering which represents the starting point for the construction of all partitions Δ-MOCK explores throughout the search. Therefore, the decoding and evaluation of this partial solution can be precomputed and exploited for the purpose of expediting the processing of individuals during optimisation.

Having completed the precomputations step, the decoding and evaluation of a new candidate individual requires processing the missing pieces of information only, *i.e.* the links not yet defined in the partial solution, which are encoded in the $|\Gamma|$-length genotype of the new representations. Processing of a $|\Gamma|$-length genotype creates new links which can merge originally separate clusters of the partial solution (Fig. 3). Such a change in the phenotype implies an amendment to the initially precomputed values of the VAR and CNN criteria.

Adjusting VAR in response to the combination of two clusters c_i and c_j requires computing the individual contribution of the new joint cluster $c_h = c_i \cup c_j$ to this measure. Rather than using (6), which becomes more computationally expensive as

[5]Notice that different clustering solutions may yield the same optimal value for the CNN criterion depending on the value chosen for parameter L.

larger and higher-dimensional problems are considered, the contribution of c_h to VAR is derived from $v(c_i)$ and $v(c_j)$, the initially precomputed contributions of c_i and c_j [47]:

$$\begin{aligned} v(c_h) &= v(c_i) + |c_i| \times \sigma(\mu_{c_i}, \mu_{c_h})^2 \\ &+ v(c_j) + |c_j| \times \sigma(\mu_{c_j}, \mu_{c_h})^2 \ , \end{aligned} \tag{9}$$

where μ_{c_h} denotes the centroid of c_h and is computed as the weighted average of the original centroids μ_{c_i} and μ_{c_j}:

$$\mu_{c_h} = \frac{(|c_i| \times \mu_{c_i}) + (|c_j| \times \mu_{c_j})}{|c_h|} \ . \tag{10}$$

Adhering to its definition in (5), VAR is recomputed by averaging the individual contributions of all the final clusters to this measure (note that (9) and (10) are generalisable and can be used when combining an arbitrary number of clusters).

Amendments to the evaluation of CNN require keeping track of the total contribution that each pair of clusters in the partial solution has made to the precomputed value for this measure. That is, a given pair of clusters c_i and c_j has contributed to the CNN criterion if and only if there exists a pair of points a and b such that: (i) at least one of these points is within the set of L nearest neighbours of the other; and (ii) one of these points belongs to c_i while the other belongs to c_j. In this way, the combination of c_i and c_j implies updating CNN by subtracting all contributions made to this measure as a consequence of the original separation of c_i and c_j.

E. Initialisation

MOCK and Δ-MOCK implement specialised initialisation routines which aim to provide these methods with a starting pool of good-quality genetic material and a close initial approximation to the Pareto front. This is sought in both cases through the generation of an initial population comprising MST-derived partitions with a range of numbers of clusters.

The original initialisation of MOCK consists of two phases. The first phase follows a DI-based strategy, similar to Δ-MOCK's approach (described below) but using a coarser-grained definition of DI [15]. The bulk of the initial population is generated during the second phase, where partitions are created by removing all MST links crossing cluster boundaries given solutions initially produced by k-means [20]. The analysis of MOCK's two-phase initialisation reveals, however, that the largest contributions to performance are achieved by the first phase, while the second phase is responsible for a significant computational overhead. It is also found that the number of clusters in the initial set of solutions is a matter which notably affects performance (the main results of this analysis are summarised in our supplementary material).

In light of this, the initialisation process is redesigned (largely drawing upon the first phase of MOCK's initialisation) as a means to improve both performance and efficiency. The precise and full new routine is sketched in Algorithm 2. It starts by including the (single-cluster) MST solution in the population (Fig. 2). All remaining individuals are iteratively created by removing the $n = k - 1$ most relevant links from the MST, endeavouring to use a different target k value at each iteration. Relevance of the MST links is determined on the basis of criterion DI (defined in Section III-B). This strategy is illustrated in Fig. 4 (with examples provided in Section V of our supplementary material).

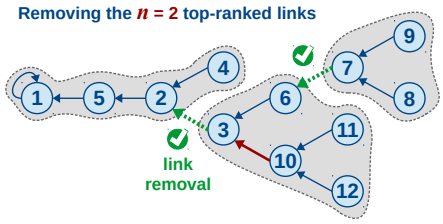

Fig. 4. Construction of an MST-derived solution by removing the $n = 2$ top-ranked MST links based on DI. This removes 2 of the 3 relevant links in set Γ (Fig. 2) and splits the MST into $k = 3$ clusters. As indicated in Section III-E, every removed MST link is replaced with a new link during the generation of MST-derived solutions. This process of link replacement is not illustrated in this figure in order to emphasise the process of link removal.

Algorithm 2 Initialisation routine of Δ-MOCK.

```
P ← {MST_solution}
while |P| < P do
    k_all ← {2, 3, ..., k_max}
    while |P| < P and |k_all| > 0 do
        choose k ∈ k_all uniformly at random
        x^Δ ← MST_derived_partition(n ← k − 1)
        P ← P ∪ {x^Δ}
        k_all ← k_all \ {k}
    end while
end while
```

Removal of MST links is only permitted for those links classified into the relevant set Γ (an example of a non-permitted link removal is provided in Fig. 4). Also, a different treatment is given to the links removed depending on the particular encoding scheme under consideration. Using the Δ-locus encoding, each MST link $i \rightarrow j$ removed is replaced with a new link $i \rightarrow h$, where h is chosen uniformly such that $h \in \{i, nn_{i1}, \ldots, nn_{iL}\} \setminus \{j\}$; *i.e.* the new link is either a self-connecting link $i \rightarrow i$ or a link from i to one of its L nearest neighbours (excluding the reintroduction of the original link $i \rightarrow j$).[6] The Δ-binary encoding does not enable the exploration of alternative links (other than the original MST links). Consequently, a link removed, emanating from i, is always replaced with a loop $i \rightarrow i$ (interpreted simply as the 'absence' of the MST link, see Fig. 3). Once the construction of an MST-derived partition is completed by means of the deletion and replacement of MST links, the resulting phenotype is translated and encoded into the corresponding $|\Gamma|$-length genotype (as explained in Fig. 3).

Parameter k_{max} defines an upper bound on the set of target k values considered during initialisation.[7] Preliminary experiments suggest that a value of about $k_{max} = 2k^*$ provides

[6]This same approach was adopted in this study when using the original (full-length) locus-based adjacency representation for comparative purposes.

[7]Notice from Algorithm 2 that if $k_{max} > P$, only a subset of k values from the set k_{all} would be required to complete the initial parent population. On the contrary, if $k_{max} < P$, multiple rounds considering the full set k_{all} are applied until completing the desired number of individuals.

the best performance, where k^* is the real (or estimated) number of clusters in the data set (refer to our supplementary material for results of these experiments). It is noteworthy that the actual number of clusters in the MST-derived solutions obtained may differ from the specific target k values employed. This is particularly the case for the Δ-locus encoding due to the above-described link replacement process. Moreover, the length of the genotype in the new encodings imposes a limit on the number of clusters a partition may involve (a partition can have, at most, as many clusters as in the partial solution defined by the fixed links in set Δ, see Fig. 2).

F. Genetic Operators

Δ-MOCK implements the same set of genetic operators as its predecessor [15]. *Uniform crossover* is employed as the genetic recombination strategy [48]. This operator takes the genotype of two parent individuals as input, and produces two offspring as output. Offspring solutions are first created as identical copies of the parents. Then, mixing of genetic material occurs based on a given recombination probability p_r. When recombination occurs, the allele of each gene can be either preserved or interchanged between the two offspring with equal probability. Thus, this operator has the ability of exploring any possible combination of genetic material from the given parents.

The *neighbourhood-biased mutation* operator is utilised [15]. When mutation is applied to a given candidate genotype, this operator determines the mutation probability individually for every gene according to the specific link that this gene currently encodes. Formally, the probability of mutating a gene encoding a link $i \rightarrow j$ is computed as:

$$p_m(i \rightarrow j) \quad = \quad \frac{1}{|\Gamma|} + \left(\frac{nn_i(j)}{|\Gamma|} \right)^2 \ , \qquad (11)$$

where $|\Gamma|$ is the length of the encoding under the new Δ-MOCK's representations (in this equation, $|\Gamma|$ is replaced with N when using the original full-length representation for comparative purposes). Computing individual mutation probabilities in this way increases the chances of discarding unfavourable links. Using the Δ-locus encoding (and similarly for the original full-length locus encoding), mutation of a link $i \rightarrow j$ implies replacing this link with a new link $i \rightarrow h$, and h will be randomly chosen from the set $\{i, nn_{i1}, \ldots, nn_{iL}\} \setminus \{j\}$ (as in the link replacement strategy adopted during initialisation, see Section III-E). The Δ-binary encoding captures the presence or absence of the original MST links only. As such, we know that a current allele value of 1 in the gene being mutated indicates that the encoded link $i \rightarrow j$ is one of the links of the MST, so that the probability of removing this link (and changing the gene's allele value to 0) is computed in the same manner as defined in (11). On the other hand, an allele value of 0 indicates that $i \rightarrow j$ is actually a self-connecting link (*i.e.* $i = j$), so that the probability of reintroducing the original MST link (and changing the gene's allele value to 1) is simply given by the first term in (11).

G. Effective Search Space

In the (full-length) locus-based adjacency representation, as originally proposed [35], every gene x_i of the genotype can conceptually encode any link $i \rightarrow j$, connecting point i to whichever point j in the data set, $i, j \in \{1, \ldots, N\}$. This results in a huge search space of size N^N. When implementing this encoding scheme in MOCK [15], however, the creation of MST-derived solutions during the initialisation process, as well as the link replacement strategy applied both during initialisation and genetic variation (the same strategy implemented in Δ-MOCK, which is described in Sections III-E and III-F), allows the algorithm to focus on a much reduced search space whose size is bounded by the following expression:

$$(L+2)^N \ . \qquad (12)$$

This is because each original MST link $i \rightarrow j$ can either be preserved, be replaced with a self-connecting link $i \rightarrow i$, or be replaced with a link from i to one of its L nearest neighbours. Therefore, a gene x_i only takes values from the set $H = \{i, nn_{i1}, \ldots, nn_{iL}\} \cup \{j\}$ throughout the evolutionary process, where $|H| = L + 1$ if point j (to which point i is originally connected in the MST) is one of i's L nearest neighbours, and $|H| = L + 2$ otherwise. This is a notable reduction of the search space, as typically $L \ll N$. In this study, we use $L = 10$, the same setting adopted in [15].

The new Δ-locus encoding can further reduce significantly the size of the search space. This depends on the setting of parameter δ, which determines the set Γ and, therefore, the length of the genotype. From the above analysis of the full-length encoding, it follows that the size of the effective search space for the ($|\Gamma|$-length) Δ-locus representation is at most:

$$(L+2)^{|\Gamma|} \ . \qquad (13)$$

Finally, the Δ-binary encoding presents the most remarkable improvement in search space reduction. By exploring only the inclusion or exclusion of the relevant MST links, the resulting size of the search space for this representation is given by:

$$2^{|\Gamma|} \ . \qquad (14)$$

The above expressions describe the accessible search space (*i.e.* the space that is reachable by the search method) under the three different representations and given the specific strategies adopted as part of the initialisation and mutation schemes. Nonetheless, the use of specialised initialisation routines in both MOCK and Δ-MOCK has important biasing effects and significantly contributes to narrowing the extent of this available search space that is actually reached during optimisation. Section V-E elaborates further on this matter.

H. Availability of Δ-MOCK

The source code of the implementations of MOCK and Δ-MOCK studied in this paper, as well as our collection of test data sets described in Section IV-C, is made available through our repository at: https://github.com/garzafabre/Delta-MOCK.

TABLE I
COMPONENTS USED IN DIFFERENT VERSIONS OF MOCK AND Δ-MOCK EVALUATED IN THIS STUDY. METHOD CONFIGURATION IS DEFINED IN TERMS OF FOUR DIFFERENT DESIGN CHOICES: (I) OPTIMISATION STRATEGY; (II) OBJECTIVE FUNCTIONS; (III) INITIALISATION ROUTINE; AND (IV) ENCODING SCHEME. THE FOUR LEFTMOST APPROACHES HAVE BEEN HIGHLIGHTED IN THIS TABLE AS THEY ARE THE MAIN FOCUS OF THE ANALYSES OF THIS PAPER. A DIFFERENT ACRONYM HAS BEEN ASSIGNED WHICH WILL BE USED TO REFER TO EACH PARTICULAR METHOD CONFIGURATION THROUGHOUT THIS STUDY. WHEN REFERRING TO APPROACHES USING THE Δ-LOCUS AND Δ-BINARY ENCODINGS, SYMBOL δ IN THEIR ACRONYMS WILL BE REPLACED BY THE SPECIFIC SETTING USED FOR THIS PARAMETER; *e.g.* IF $\delta = 80$, APPROACHES Δ^L_δ AND Δ^B_δ WILL BE REFERRED TO AS Δ^L_{80} AND Δ^B_{80}, RESPECTIVELY.

	MOCK or Δ-MOCK version							
Components	$\mathbf{M}^{H07}$	$\mathbf{M}^*_V$	$\boldsymbol{\Delta}^L_\delta$	$\boldsymbol{\Delta}^B_\delta$	$\boldsymbol{L}_\delta$	$\boldsymbol{B}_\delta$	$\boldsymbol{R}^L_\delta$	$\boldsymbol{R}^B_\delta$
Optimisation:								
PESA-II	•							
NSGA-II		•	•	•	•	•	•	•
Objectives:								
ODV	•							
VAR		•	•	•	•	•	•	•
CNN	•	•	•	•	•	•	•	•
Initialisation:								
MOCK	•	•			•	•		
Δ-MOCK			•	•				
Random							•	•
Encoding:								
Locus (full-length)	•	•						
Δ-locus			•		•		•	
Δ-binary				•		•		•

TABLE II
SUMMARY OF THE MAIN PARAMETER SETTINGS USED IN THIS STUDY.

Population size	$P = 100$
Number of generations	$G_{max} = 100$
Recombination probability	$p_r = 1.0$
Mutation probability	$p_m(i \to j) = \frac{1}{\lvert\Gamma\rvert} + \left(\frac{nn_i(j)}{\lvert\Gamma\rvert}\right)^2$
Maximum k (initialisation)†	$k_{max} = 2k^*$
Neighbourhood parameter	$L = 10$
Encoding-length parameter‡	$\delta \in \{0, 50, 80, sr5, sr2, sr1\}$

† Where k^* denotes the real number of clusters in our test data sets.
‡ Where $sr\eta = 100 - \left(100 \times \eta/\sqrt{N}\right)$, for $\eta \in \{5, 2, 1\}$.

IV. EXPERIMENTAL SETUP

This section summarises the approaches evaluated and compared later in Section V, and describes the data sets, settings and experimental conditions adopted during this study.

A. MOCK and Δ-MOCK

Table I outlines the configuration of different variants of MOCK and Δ-MOCK studied in this paper. These variants present specific choices regarding the implemented search strategy, optimisation criteria, initialisation routine, and genetic representation. Approaches M^{H07}, M^*_V, Δ^L_δ, and Δ^B_δ are our main subjects of analysis, while the remaining configurations have been considered with the aim of evaluating specific algorithmic decisions and behaviours.

M^{H07} corresponds to the original version of MOCK reported in [15]. M^*_V incorporates changes to the search strategy and optimisation criteria, but preserves the original initialisation routine and (full-length) representation as in M^{H07}. In this study, M^*_V serves as the primary baseline for investigating the suitability of Δ-MOCK approaches Δ^L_δ and Δ^B_δ (using the new Δ-locus and Δ-binary reduced-length encodings, respectively). This is because M^*_V uses the same search strategy, optimisation criteria, and is implemented within the same framework as Δ^L_δ and Δ^B_δ. As such, the use of M^*_V as the baseline (instead of M^{H07}) enables a fair analysis of the computational efficiency of the methods and makes it possible to compare the characteristics of the PFAs obtained. Moreover, evaluation with respect to M^*_V allows us to assess the impact of the new initialisation and representation schemes which are the main innovative elements of Δ-MOCK.

Finally, approaches L_δ, B_δ, R^L_δ, and R^B_δ are included in this study to evaluate the relevance of the new reduced-length representations without relying on the specialised initialisation used by Δ^L_δ and Δ^B_δ. On the one hand, configurations L_δ and B_δ use the original initialisation routine of MOCK [15], see Section III-E. On the other hand, approaches R^L_δ and R^B_δ implement a random initialisation as follows: the allele of a given gene, which will encode a link emanating from data point i, is selected uniformly from the set $\{i, nn_{i1}, \ldots, nn_{iL}\}$ if using R^L_δ, or from the set $\{0, 1\}$ if using R^B_δ.

The parameter settings adopted for all the above-described variants of MOCK and Δ-MOCK, except M^{H07}, are summarised in Table II. For M^{H07}, we use the same settings as reported in [15]. Note from Table II the use of a setting of $k_{max} = 2k^*$ for Δ-MOCK's initialisation strategy; this setting was determined based on preliminary testing (results are included in our supplementary material). To perform a fair comparative analysis, an equivalent range of target k values is used for all approaches based on the original initialisation routine of MOCK (M^{H07}, M^*_V, L_δ, and B_δ). Also, note from the table that we explore six different settings for parameter δ, which achieve different genotype lengths for the Δ-locus and Δ-binary encodings. Three of such settings, namely $sr5$, $sr2$, and $sr1$, set the value of δ individually for each given problem instance so that the resulting length of the encoding is defined as a function of $\sqrt{N}$; more specifically, settings $sr5$, $sr2$, and $sr1$ (as defined in Table II) lead to encoding lengths of $\sim 5\sqrt{N}$, $\sim 2\sqrt{N}$, and $\sim\sqrt{N}$, respectively. Through the consideration of such settings, this study investigates the extent to which we can employ encoding lengths that do not grow linearly with the size of the problem N, thus providing highly significant reductions in the size of the search space and considerable improvements in terms of computational efficiency.

B. Reference Methods

While the primary goal of this paper is to investigate the advantages of Δ-MOCK with respect to its predecessor, MOCK, results for additional clustering methods from the literature are also reported as a reference.

1) k-Means: The well-known partitioning method k-means [20] is run for a diverse set of values of k. This results in candidate solution sets which are more comparable to those produced by MOCK and Δ-MOCK. A single run of Δ-MOCK can generate a solution set with as many different numbers of clusters k as the population size, P. Given the setting adopted $P = 100$ (see Table II), a total of 100 different k values around k^* is considered during the experiments with k-means. More specifically, for a data set with k^* clusters, k-means is run for all k values from the set $\{ \max\{2, k^* - 50\}, \ldots, k^*, \ldots, \max\{2, k^* - 50\} + 99 \}$.

For each target k considered, k-means is run repeatedly (a total of 50 times), each time starting from a randomly generated initial partition. From the set of all repetitions, the solution with the best performance, based on the intra-cluster variance criterion, is selected and included in the final solution set. This is done as a means of accounting for suboptimality. In all the cases, a maximum number of 100 iterations is used as the stopping criterion for the k-means procedure (the iterative process is stopped earlier if the algorithm converges).

2) Multiobjective Genetic Algorithm for Clustering: This method, referred to as MGA in our comparative analysis, was proposed in [21] and further investigated in [49]–[51]. MGA is based on NSGA-II [46], uses a centroid-based representation, and optimises simultaneously the Xie-Beni (XB) index [52] and the fuzzy C-means (J_m) [53] measure.[8] Two additional variants of this method are explored: MGA_{VL} and MGA^*_{VL}. MGA_{VL} implements a variable-length encoding which enables the generation of partitions with varying k. This variable-length encoding has been utilised in other studies by the same authors [54], [55]. MGA^*_{VL} is our own extension of MGA_{VL} that uses the same optimisation criteria and an MST-based initialisation scheme as in Δ-MOCK. We include this as an additional comparison, as it enables a more direct comparison of the centroid-based and graph-based representations, the main conceptual difference between MGA^*_{VL} and Δ-MOCK.

In all the cases, a population of size $P = 100$ and $G_{max} = 100$ generations are considered (this is consistent with the number of solution evaluations performed by MOCK and Δ-MOCK). The recombination and mutation probabilities are respectively set to $p_r = 0.8$ and $p_m = 1/l$ (l being the encoding length) as used in [21]. In MGA_{VL} and MGA^*_{VL}, the maximum number of clusters is set to $2k^*$, which also sets the maximum possible length of the encoding. Finally, the initialisation scheme of MGA^*_{VL} generates MST-derived solutions following the same strategy as in Δ-MOCK (Section III-E). In order to construct the initial population of centroid-based genotypes, cluster centres are initialised to the coordinates of randomly selected points from the connected components (clusters) in the corresponding MST-derived partitions.

C. Experimental Data

A total of 358 data sets are considered for the experiments of this study. Eight large two-dimensional data sets have been constructed using real, open data about street-level crime in the United Kingdom (UK).[9] Attributes in these data sets correspond to the geographic coordinates (latitude and longitude) of crime reports, and crimes are clustered according to the individual police forces in the UK. The size of these problems is $\sim$30,000 data points (see Table III in Appendix A). Data sets were constructed by selecting different subsets of the police forces, presenting varying levels of difficulty. These data sets will be denoted by UKC1, UKC2, ..., UKC8.

The remaining 350 problems are synthetic data sets generated using the ellipsoidal generator previously used during the evaluation of MOCK [15]. This generator creates clusters with arbitrary elongation and orientation, whose positions are then optimised using a genetic algorithm in order to arrange the clusters in a compact configuration. As shown in Table III (Appendix A), these data sets present varying sizes, ranging from $\sim$2,500 (in average) to over 9,000 data points. Data sets are organised into 35 problem configurations based on the dimensionality of the data (d) and number of clusters (k^*): $d \in \{20, 50, 100, 150, 200\}$ and $k^* \in \{10, 20, 40, 60, 80, 100, 120\}$. A set of 10 random instances were generated for each problem configuration. A specific problem configuration will be referred to as xd-yc throughout this study, where $x = d$ and $y = k^*$.

The reader is referred to our supplementary material for further information about the data sets employed in this paper.

D. Performance Assessment

For all the clustering methods analysed and compared in this study, a total of 21 independent executions for each problem instance were performed. Different performance indicators and tools are employed to evaluate the results of these methods from different perspectives, as described below.

1) Clustering Performance: The *Adjusted Rand Index* (ARI) measure is used as the main indicator of clustering performance [56]. Given two partitions A and B for a collection of data points, this measure determines the similarity of A and B by analysing the pairwise co-assignment of points between the two partitions. ARI is defined in the range $[\sim 0, 1]$; the larger the value for ARI, the better the correspondence between A and B. If we let A be a candidate partition generated by a given clustering method, and we let B be the true partition (ground truth) of the data (which is known for our test data sets), ARI can thus provide an objective estimate of the method's ability to identify the inherent cluster structure of the data.[10] Each independent execution of the clustering methods studied here produces a set of candidate partitions. In all the cases, only the best solution from this set, according to the ARI measure, is selected and considered in the results reported in Section V; hence, we report statistics of the best solution qualities produced by (*i.e.* contained in the candidate solution sets of) the different methods analysed.

[8]The XB and J_m criteria are originally defined for fuzzy clustering. The corresponding adaptations for crisp clustering are considered here.

[9]This data is available at: https://data.police.uk/

[10]We say that this is an objective assessment of clustering performance since the methods evaluated in this study, as unsupervised approaches, do not exploit this external knowledge of the true partition during optimisation. Furthermore, ARI is not biased towards algorithms employing a particular type of clustering objective (as would be the case for measures of cluster validity that consider the intrinsic structure of the data set).

2) Quality of the Pareto Front Approximations: Such as for any multiobjective optimiser, it is essential to investigate the characteristics of the PFAs obtained as a means to understand the performance and behaviour of MOCK and Δ-MOCK. Our analysis in this respect focuses on the visualisation of the differences between the (first-order) *empirical attainment functions* (EAFs) of different method configurations [57], [58]. An EAF represents an estimate of the probability that an arbitrary objective vector is *attained*[11] during a single execution of a given method. Such an estimate is computed from the frequency with which the vector is attained by the method in the set of all independent executions performed. EAFs thus provide a pictorial representation which captures an algorithm's behaviour based on its outcomes. By visualising the differences between the EAFs of two method configurations it becomes possible to identify whether, and in which particular regions of the objective space, a configuration performs better than the other. The plots of the EAFs included in this paper are generated using the tools reported in [58]. In all the cases, objective values are normalised to the range $[0, 1]$ and then, for visualisation purposes, replaced with their square roots in order to amplify the differences between the approaches compared.

In addition, two performance indicators from the EMO literature are adopted in this study: the *hypervolume indicator* (HV) [59], [60] and the *inverted generational distance indicator with modified distance calculation* (IGD$^+$) [61]. Both HV and IGD$^+$ are capable of expressing the quality of a PFA with regard to both extent and proximity to the true Pareto front. These indicators are computed in this study after normalising objective values to the range $[0, 1]$. HV is a Pareto-compliant indicator that measures the portion of the objective space (bounded by a reference objective vector $\mathbf{r}$) which is dominated by a given PFA. Here, the reference point is set to $\mathbf{r} = [1.01, 1.01]^T$ in all the cases given the use of normalised objective vectors. Higher HV values are always preferred. The IGD$^+$ indicator is weakly Pareto-compliant; broadly, this indicator uses a reference set to represent the true Pareto front, and is computed as the average (modified) distance from each point in this set to the closest point in the PFA obtained. In this study, the reference set comprises the overall best known PFA for each individual problem instance, and is computed by merging the PFAs obtained by all of the method configurations analysed here and then removing all dominated vectors. Contrary to HV, IGD$^+$ is to be minimised.

3) Computational Efficiency: The suitability of Δ-MOCK is also explored from the standpoint of its computational efficiency. This is evaluated specifically with respect to MOCK configuration M^*_V, which is used as the baseline as stated in Section IV-A. Both M^*_V and Δ-MOCK are implemented within the same framework, sharing most of the code which define their functionality. This enables a fair comparative analysis in this context. These approaches are implemented in *C++11*, and are compiled with *Clang* using the '*-O3*' optimisation flag for the experiments of this study. Execution times are measured (internally) using class '*std::chrono::steady_clock*'. All the experiments of this study are run on a (exclusively dedicated) computer with a 3.5 GHz 6-Core Intel Xeon E5 processor and 32GB of RAM.

4) Statistical Significance Analysis: Hypothesis testing is conducted to investigate whether the differences observed between the performance of specific method configurations are statistically significant or not. Method configurations are analysed pairwise using the (non-parametric) *Mann-Whitney U test*, considering a significance level of $\alpha = 0.05$ in all the cases. Only a certain subset of the method configurations are considered in order to minimise the number of inferences that are made based on the same samples. In addition, *Bonferroni correction* is applied to account for multiple testing issues.

[11] A vector $\mathbf{w}$ in objective space is said to be attained by the method if and only if the PFA produced includes either $\mathbf{w}$ or any vector $\mathbf{z}$ such that $\mathbf{z} \prec \mathbf{w}$.

V. Results

This section presents the results of a series of experiments conducted to investigate the suitability and understand the functioning and behaviour of Δ-MOCK. The analysis of these results is organised as follows. First, the overall evaluation of specific Δ-MOCK configurations in terms of clustering performance is presented in Section V-A. Then, Section V-B complements this evaluation by analysing and comparing the convergence behaviours of Δ-MOCK and MOCK. Section V-C contrasts the characteristics of the PFAs obtained by these approaches. Δ-MOCK's advantages from the perspective of computational efficiency are illustrated in Section V-D. Finally, the role of the new reduced-length encodings of Δ-MOCK, as well as the setting of parameter δ and its interaction with the initialisation strategy, is studied in Section V-E.

A. Clustering Performance

This section investigates the ability of Δ-MOCK to produce high-quality clustering solutions. Δ-MOCK configurations Δ^L_{sr5} and Δ^B_{sr5}, which result from the use of a setting of $\delta = sr5$ (alternative settings are explored later in Section V-E), are evaluated and compared with respect to the baseline MOCK configuration M^*_V. In addition, the original implementation of MOCK (denoted by M^{H07}) and the k-means and MGA^*_{VL} methods are included in this comparison as a reference. The results of this analysis are summarised in Fig. 5, with more detailed results and the corresponding statistical significance analysis presented in Table III (Appendix A).

As expected, all of the methods evaluated scored very competitive results for most of the UKC data sets. The particular characteristics of these problems (refer to our supplementary material for a visualisation of these two-dimensional data sets) allowed the centroid-based methods k-means and MGA^*_{VL} to closely approximate the correct cluster structure of the data in most of the cases. Also, the use of k-means as part of the initialisation routine of MOCK (M^{H07} and M^*_V) is seen to provide an advantage over Δ-MOCK (Δ^L_{sr5} and Δ^B_{sr5}) under these conditions. It is evident, however, that as the difficulty of the problem increases, and particularly when considering the synthetic data sets which involve elongated clusters, the

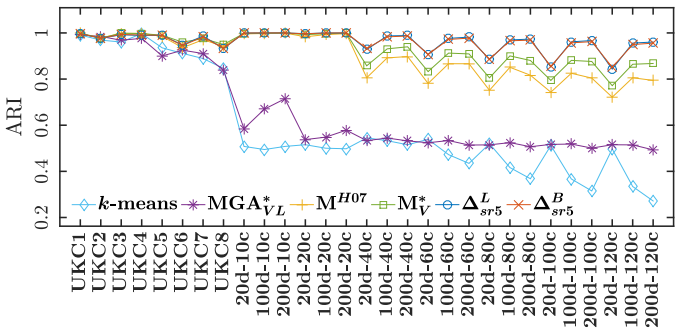

Fig. 5. Summary of the results from the perspective of clustering performance. Curves show the average ARI scored by MOCK/Δ-MOCK approaches M^{H07}, M^*_V, Δ^L_{sr5}, and Δ^B_{sr5}, and the contestant methods k-means and MGA^*_{VL}. Results are shown for all UKC data sets and a subset of the synthetic problem configurations. Refer to Table III in Appendix A for detailed results.

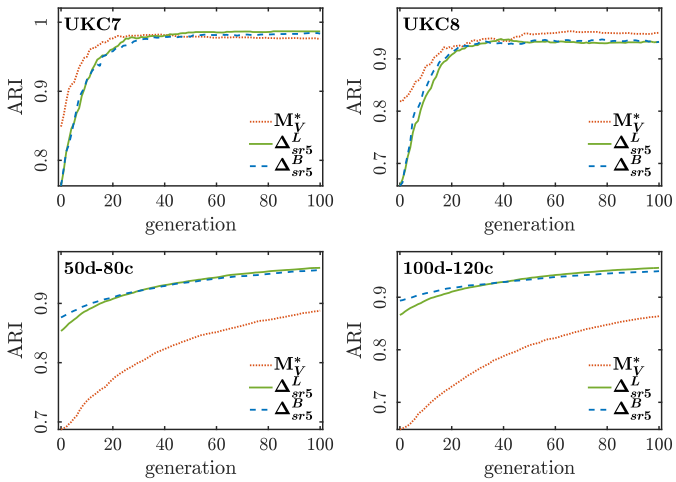

Fig. 6. Convergence plots indicating the highest ARI in the population at every generation of the evolutionary process. Plots contrast the convergence curves of approaches M^*_V, Δ^L_{sr5}, and Δ^B_{sr5} in two UKC problems and two synthetic data set configurations (average results for all instances and repetitions).

inability of centroid-based approaches to capture arbitrary cluster shapes provokes an important drop in performance.[12]

Methods using a graph-based representation (M^{H07}, M^*_V, Δ^L_{sr5}, and Δ^B_{sr5}) consistently report better results than the centroid-based approaches in all of the synthetic problem configurations (and in the most difficult UKC problems). Approach M^*_V presents a notable improvement with respect to the original version of MOCK, M^{H07}. Such an improvement stems mainly from the change in the underlying search engine of MOCK.[13] M^*_V uses NSGA-II as a means of accounting for the high selection pressure observed in the original implementation based on the strongly elitist PESA-II (see Section III-A). The fact that the new implementation based on NSGA-II shows an increased overall performance confirms that a strategy with less selection pressure is advantageous in this specific scenario and allows us to better exploit the highly-optimised genetic material introduced during initialisation.

The use of the new initialisation scheme and the new reduced-length representations, which gives rise to Δ-MOCK approaches Δ^L_{sr5} and Δ^B_{sr5}, leads to a further boost in clustering performance. Approaches Δ^L_{sr5} and Δ^B_{sr5} are found to outperform the baseline M^*_V, achieving statistically significant improvements in the ARI measure for all of the synthetic problem configurations. Using the Δ-locus encoding, Δ^L_{sr5} tends to produce better results than Δ^B_{sr5} which uses the Δ-binary encoding (see Table III). The performance differences between the two encodings are found to be statistically significant for most of the synthetic problems with $k^* \geq 40$ clusters.

It is worth remarking that, given the setting adopted $\delta = sr5$, Δ^L_{sr5} and Δ^B_{sr5} operate on genotypes of a substantially reduced length in comparison to M^*_V. This, together with the use of a more efficient initialisation routine, brings additional benefits by decreasing the computational effort as evaluated in Section V-D. The remaining subsections complement the analysis presented here, and attempt to illustrate the specific roles of the new initialisation and representation schemes and how they contribute, individually and collectively, to Δ-MOCK's performance and computational efficiency.

[12]It is interesting to see from Table III that, despite its low overall performance, MGA^*_{VL} is found to provide much better results than approaches MGA and MGA_{VL} in most of the cases. This highlights the benefits of using a representation which can encode partitions with different numbers of clusters, as well as the advantages of employing a specialised initialisation routine and a more appropriate selection of optimisation criteria.

[13]Besides using a different search method, M^*_V changes one of MOCK's optimisation criteria, replacing ODV with VAR. This change, however, is not found to affect (neither positively nor negatively) the algorithm's performance (results not shown). As stated in Section III-C, this adaptation does not seek to alter the algorithm's behaviour, but seeks to exploit the advantages of the VAR criterion in the context of the delta-evaluation feature of Δ-MOCK.

B. Convergence Behaviour

This section delves into the convergence behaviour of MOCK and Δ-MOCK, aiming to generate a better understanding of the differences observed in Section V-A between the clustering performance of approaches M^*_V, Δ^L_{sr5}, and Δ^B_{sr5}.

Fig. 6 illustrates the ability of approaches M^*_V, Δ^L_{sr5}, and Δ^B_{sr5} to discover good-quality partitions (as captured by the ARI measure) throughout the evolutionary process. It is interesting to observe that, for all the three approaches analysed, the simultaneous optimisation of the VAR and CNN criteria implicitly and effectively leads to the optimisation of clustering quality (ARI gradually increases with the progress of the search process). Note that ARI is computed for performance assessment only, and is never exploited to guide optimisation as indicated in Section IV-D. This provides corroborating evidence of the suitability of the clustering criteria adopted as objective functions and of the conceptual advantages of the multiobjective approach to data clustering in general.

Fig. 6 exhibits marked differences in the clustering quality that the three approaches report for the initial population (generation 0 in the plots). On the one hand, the plots for the UKC data sets confirm that, as discussed in Section V-A, the use of k-means within the initialisation strategy of M^*_V offers a competitive advantage over Δ^L_{sr5} and Δ^B_{sr5} given the characteristics of these problems. Note, however, that this is rapidly compensated by the ability of Δ^L_{sr5} and Δ^B_{sr5} to perform a more-focused exploration in the substantially smaller solution space of their reduced-length representations. On the other hand, the results for the synthetic problems

also confirm that such a competitive advantage of MOCK's initialisation based on k-means does not hold in scenarios with more challenging (*e.g.* elongated) cluster shapes. In these scenarios, the new initialisation of Δ-MOCK is shown to achieve a remarkable increase in clustering quality, thus being a major contributor to the method's performance.

The plots for the synthetic data sets illustrate clear differences between the convergence behaviour of Δ-MOCK configurations Δ^L_{sr5} and Δ^B_{sr5}. Whilst Δ^B_{sr5} shows better results at the beginning of the search, Δ^L_{sr5} tends to reach higher ARI values in the end.[14] This reflects the key conceptual differences between the representations used by the two approaches. The Δ-binary encoding (used by Δ^B_{sr5}) works by specifying whether each of the (relevant) MST links is retained or removed; as such, this approach explores more explicitly the separation of the MST into connected components. During initialisation, thus, the Δ-binary encoding is able to produce partitions with every given target k value considered (see Section III-E). This results in a high diversity of good-quality initial individuals which may explain the better ARI values observed for this approach at the initial stages of the search process. In contrast, the Δ-locus encoding (used by Δ^L_{sr5}) permits the replacement of the MST links with other alternative links between data points; not only can these alternative links reconnect the same MST components originally separated through the removal of the MST link, but also can connect components which are not directly linkable by any of the MST links. In this way, and despite yielding a lower initial performance, such a link replacement strategy grants the Δ-locus encoding access to (potentially relevant) areas of the solution space, which are not available to the Δ-binary encoding. This offers an explanation for the better performance shown by Δ^L_{sr5} at the end of the search process.

C. Pareto Front Approximations

Whereas the suitability of Δ-MOCK has hitherto been demonstrated from the perspective of clustering performance (see Sections V-A and V-B), this section seeks to further investigate and explain the advantages of this method from the perspective of the characteristics of the PFAs obtained.

Fig. 7 exemplifies the contrasting characteristics found in the PFAs of MOCK (M^*_V) and Δ-MOCK (Δ^L_{sr5} and Δ^B_{sr5}). By analysing the differences between the EAFs of these approaches, it is clear that M^*_V presents the tendency of reaching more deeply into the low-VAR (high-CNN) regions of the objective space; this behaviour seems to be more evident as k^* increases, and especially in comparison to Δ^L_{sr5}. This result can be explained by the fact that VAR is relatively simple to optimise, as the value of this criterion naturally decreases with the increase in the number of clusters (k) in the partition evaluated (see Section III-C). The full-length representation of M^*_V is able to encode partitions with high k (indeed, as high as N), in contrast to the reduced-length representations of Δ^L_{sr5} and Δ^B_{sr5} which are only capable of encoding partitions with a limited k (and with a certain minimum reachable value for VAR).[15] Accordingly, Fig. 8 confirms that the PFAs obtained

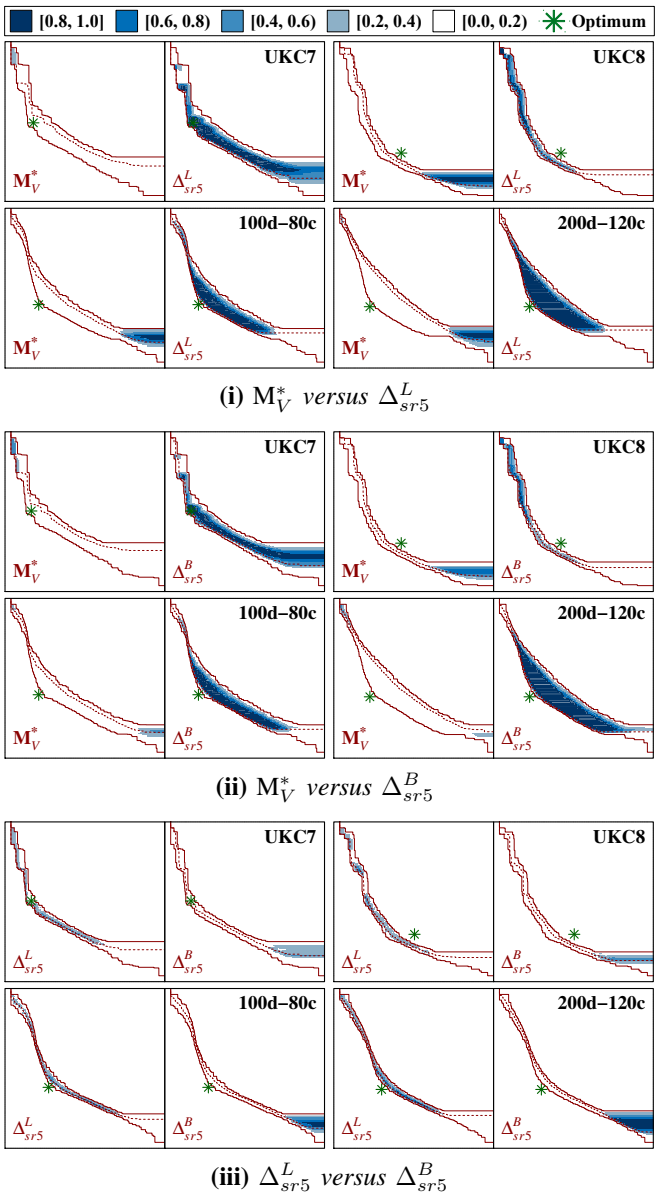

Fig. 7. Comparison of the EAFs computed from the PFAs obtained from all independent executions of MOCK/Δ-MOCK approaches M^*_V, Δ^L_{sr5}, and Δ^B_{sr5}. Approaches are compared pairwise, and results are illustrated for problems UKC7, UKC8, and individual instances of synthetic problem configurations 100d-80c and 200d-120c. In the plots, the x-axis and y-axis denote respectively the (normalised) CNN and VAR criteria. Each plot highlights the differences in the point attainment probabilities which are in favour of each individual approach in the pairwise comparison. The magnitudes of such differences are encoded using different intensities of blue colour: the darker the blue, the larger the difference (see legend at the top). Solid lines represent the grand best (lower line) and grand worst (upper line) attainment surfaces (common to the two methods compared), and the dashed line denotes the median attainment surface (specific to each individual approach). In addition, each plot includes a marker to illustrate the location of the optimum (real clustering of the test data set) in the objective space.

[14]The different slopes in the convergence curves of Δ^L_{sr5} and Δ^B_{sr5} (see Fig. 6) suggest that even more meaningful performance differences (in favour of Δ^L_{sr5}) might be observed if longer runs of these methods are considered.

[15]Depending on the setting of δ, a potentially large subset of the MST links are fixed for all phenotypes, and the partial solution defined by such fixed links (Fig. 2) sets the maximum k and the minimum VAR that can be reached. During optimisation, the clusters defined by the partial solution are combined, which can only lead to the decrease of k and the increase of VAR.

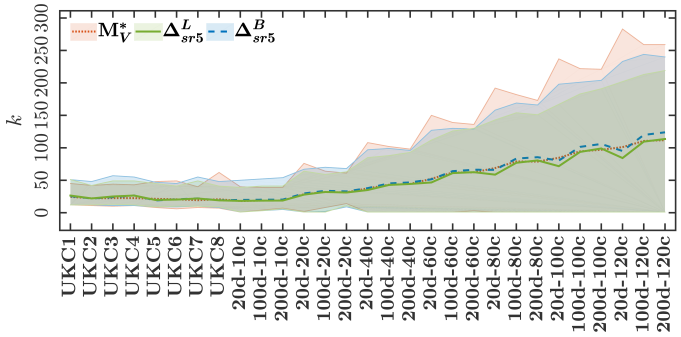

Fig. 8. Number of clusters (k) in the partitions contained in the PFAs produced by M_V^*, Δ_{sr5}^L, and Δ_{sr5}^B. Curves show the arithmetic mean and shaded areas show the full range of k values covered. Results are shown for all UKC data sets and a subset of the synthetic problem configurations.

by M_V^* tend to contain partitions with higher k values than those of the Δ_{sr5}^L and Δ_{sr5}^B approaches (including k values substantially exceeding k^* and which, in some cases, might be considered beyond practical relevance). Correlating Figs. 7 and 8, it is interesting to note that the higher the maximum k values produced by a method, the better is found to be this method with regard to the optimisation of VAR.

The EAFs of approaches Δ_{sr5}^L and Δ_{sr5}^B indicate that the improvements in the initialisation routine and the reduction in the length of the encoding (and corresponding search space) allow Δ-MOCK to produce approximations sets of much greater quality. In comparison to M_V^*, both Δ_{sr5}^L and Δ_{sr5}^B exhibit an increased convergence ability towards the central regions of the Pareto frontier. Such central regions, presenting better compromises between the VAR and CNN criteria, are not only more challenging to reach, but can also be assumed of primary interest since they usually correlate with the location of the optimal clustering solution (see Fig. 7). Improved convergence towards these regions of the objective space consequently translates into an increased clustering performance in most cases as observed in Sections V-A and V-B. The enhanced convergence capabilities of Δ-MOCK are further highlighted by the HV and IGD^+ performance indicators; as shown in Table IV (Appendix A), Δ_{sr5}^L and Δ_{sr5}^B perform significantly better than M_V^* under these indicators in most of the cases.

Finally, the comparison of the EAF of Δ_{sr5}^L and Δ_{sr5}^B emphasises the conceptual differences between the two encodings of Δ-MOCK. As discussed in Section V-B, the Δ-binary encoding used by Δ_{sr5}^B is more effective at exploring the separation of the MST into connected components. This produces PFAs involving partitions with higher k values in all the problems considered (see Fig. 8), which helps explain the fact that Δ_{sr5}^B surpasses Δ_{sr5}^L with respect to the optimisation of the VAR criterion. This fact is also evidenced by the better scores that Δ_{sr5}^B obtains for the HV indicator in most of the cases, see Table IV. On the other hand, the link replacement strategy used for the Δ-locus encoding allows Δ_{sr5}^L to achieve a slight convergence improvement towards central regions of the Pareto front. These slight but still important differences are reflected in better results for the IGD^+ indicator in the majority of the cases and account for the superiority that Δ_{sr5}^L has shown in terms of clustering performance.

D. Computational Effort

Initial analyses of the computational efficiency of MOCK revealed that, in average for the clustering problems considered here, about 80% of the total computational costs of this method are derived from the optimisation process.[16] This comprises the costs of initialisation (generation of the initial population) and the costs of the main optimisation cycle which are mostly related to the decoding, evaluation, and general handling of candidate individuals. In light of this, Δ-MOCK's initialisation and encoding schemes are designed to target the computational overhead caused by these processes. This section aims to illustrate the accomplishments of Δ-MOCK in this respect.

Fig. 9 summarises the execution times reported by Δ-MOCK approach Δ_{sr5}^L during the optimisation process and contrasts them with respect to those of the baseline MOCK configuration M_V^*.[17] It is important to remark at this point that, in all the cases, both Δ_{sr5}^L and M_V^* have been run using the same population size and number of generations (refer to Section IV-A and Table II for details). Therefore, Δ_{sr5}^L and M_V^* performed the same number of solution evaluations, and the differences in the execution times observed in Fig. 9 are due to the benefits of the new initialisation, representation, and preprocessing schemes of Δ-MOCK.

According to Fig. 9, Δ_{sr5}^L is able to reduce $\sim 96.3\%$ of the total optimisation time, in average, with respect to M_V^*. Δ-MOCK's gains in terms of computational performance become more evident with the increase in problem size and dimensionality. The new initialisation routine certainly contributes to computational efficiency in an important manner. As can be seen from the figure, Δ-MOCK almost completely removes ($\sim 99.4\%$ savings in average) the expenses involved with the original initialisation mechanism of MOCK (particularly related to the use of k-means, which becomes more expensive for larger and higher-dimensional problems).

Finally, by turning our attention to the times of the main optimisation cycle it is possible to isolate the advantages that the new reduced-length encodings provide in this context. Evidently, these advantages are dependent on the setting of parameter δ. As Fig. 9 indicates, Δ-MOCK removes over 93% (in average) of the computational effort during the main optimisation cycle, given the value of $\delta = sr5$ chosen for this experiment. Such a setting of parameter δ allows Δ-MOCK to use significantly shorter (more efficient to handle) genotypes (see Section V-E). Moreover, this allows the precomputation of a substantial amount of information which is later exploited to speed up the decoding and evaluation of all candidate solutions during the search process.[18] Section V-E complements this analysis by exploring alternative settings for parameter δ.

[16] The remaining $\sim 20\%$ of the costs are related to the tasks of data loading and initial precomputations (distance matrix, nearest neighbours, MST).

[17] Both Δ_{sr5}^L and Δ_{sr5}^B report comparable execution times. Hence, only the results for Δ_{sr5}^L are analysed here for convenience.

[18] Such precomputations involve the decoding and evaluation of a (single) partial solution, see Section III-D, the computational cost of which is already considered as part of the costs of the initialisation reported in Fig. 9.

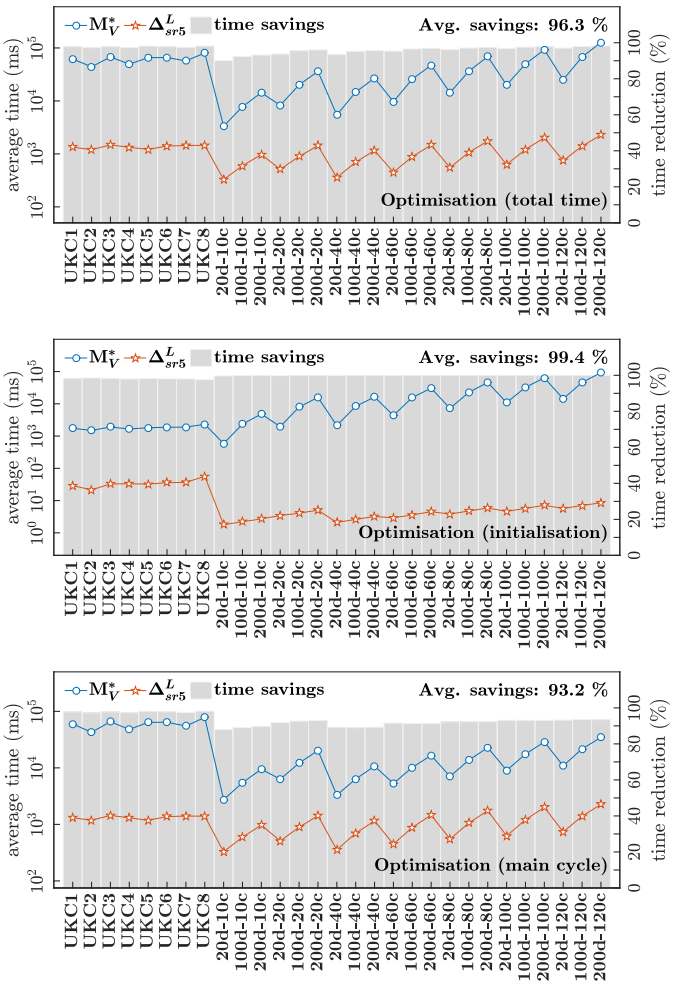

Fig. 9. Execution times scored by Δ^L_{sr5} and the baseline M^*_V. Results are shown regarding the total time of the optimisation (top), the time of the initialisation routine (middle), and the time of the main optimisation cycle (bottom). In the plots, curves show the average time in milliseconds (left y-axis, which is shown in logarithmic scale) for Δ^L_{sr5} and M^*_V. In addition, bars show the average time savings achieved by Δ^L_{sr5} (right y-axis) with respect to M^*_V: letting a be the time of Δ^L_{sr5} and b be the time of M^*_V, savings are computed as $100 \times (b-a)/b$. Results are shown for all UKC data sets and a subset of the synthetic problem configurations. The average time savings achieved across all problems are indicated in the top-right corner of each plot.

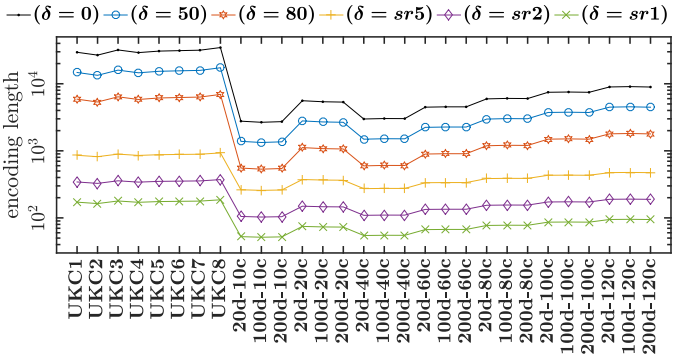

Fig. 10. Encoding length resulting from the use of different settings for parameter δ. Settings considered: $\delta \in \{0, 50, 80, sr5, sr2, sr1\}$. Results are shown for all UKC data sets and a subset of the synthetic problem configurations. For the synthetic problems, the average encoding length considering all 10 instances is shown. Note that results are presented in logarithmic scale.

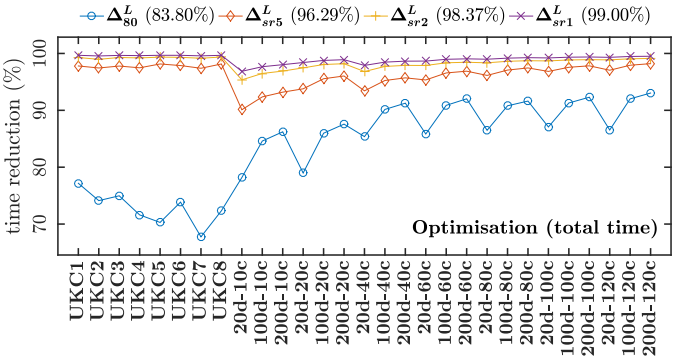

Fig. 11. Impact of parameter δ in computational efficiency. Average time savings achieved by Δ^L_δ in all UKC data sets and a subset of the synthetic problem configurations. Total time of the optimisation process is considered (lines 3 to 9 in Algorithm 1), and savings are computed with respect to the baseline MOCK configuration M^*_V as in Fig. 9. Settings explored: $\delta \in \{80, sr5, sr2, sr1\}$. For each of these settings, the legend at the top of the figure shows the average savings across all problems considered.

E. Encoding Length and its Interplay with Initialisation

This section investigates more thoroughly the relevance of the reduced-length representations of Δ-MOCK, and the role of parameter δ in determining the advantages they provide in terms of computational efficiency and clustering performance. Furthermore, preliminary analyses have pinpointed a clear interaction between the representation and initialisation schemes. This section aims to explore and illustrate this interaction.

Six different settings for parameter δ are considered in this analysis: $\delta \in \{0, 50, 80, sr5, sr2, sr1\}$. Settings $sr5$, $sr2$, and $sr1$, which set the value of δ and define the length of the encoding as a function of $\sqrt{N}$ (see Section IV-A), are included in this study in order to address the important question of how far we can go in reducing the length of the encoding without compromising the method's effectiveness. As shown in Fig. 10, these settings use only a fraction of the encoding length used by the original representation (or, equivalently, $\delta = 0$). More specifically, the resulting encoding length for settings $sr5$, $sr2$, and $sr1$ is found to be, respectively, only about 7.1%, 2.8%, and 1.4% (in average) of the original encoding length for our collection of test data sets. Intuitively, the shorter the length of the encoding, the more meaningful the benefits we can obtain in terms of computational efficiency as indicated in Fig. 11. It is possible to observe from Fig. 11 that the increase in the value of δ reliably translates into an increase in the computational efficiency of Δ-MOCK. In average, time reductions range from ∼84%, when using $\delta = 80$, to ∼99%, when using $\delta = sr1$, with respect to the baseline MOCK configuration M^*_V (which uses a full-length representation).

Despite these indisputable advantages from the standpoint of computational efficiency, it is not yet clear whether the significant pruning of the search space achieved by the reduced-length encodings can also have a determining effect on clustering performance. This is especially true given that, so far in this paper, the new reduced-length representations have not been investigated in isolation, without using Δ-MOCK's specialised initialisation routine. We hereby extend our study and evaluate these representations in combination with three

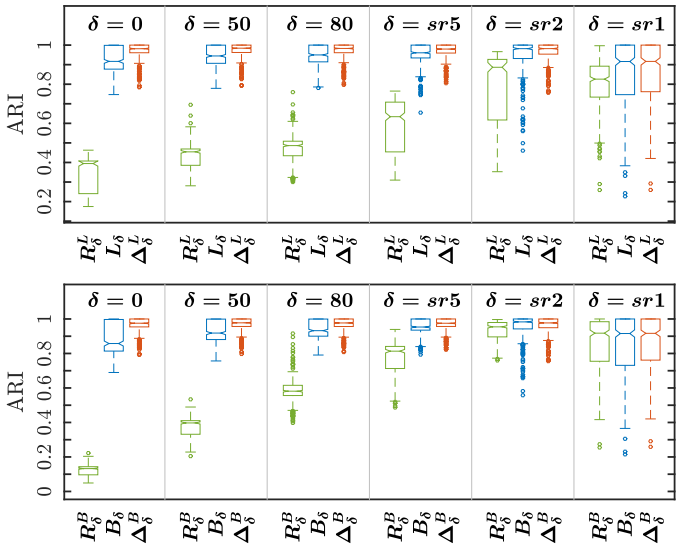

Fig. 12. Impact of the reduced-length encodings and parameter δ in terms of clustering performance. Results for the Δ-locus and Δ-binary encodings (top and bottom, respectively) consider the following settings: $\delta \in \{0, 50, 80, sr5, sr2, sr1\}$. The figure contrasts the results obtained when using: (i) a random initialisation, R^L_δ and R^B_δ; (ii) the original initialisation of MOCK [15], L_δ and B_δ; and (iii) the new specialised initialisation, Δ^L_δ and Δ^B_δ. Box plots summarise the results (average ARI) obtained across all instances and runs performed for all 35 synthetic problem configurations.

different initialisation schemes: (i) a random initialisation scheme; (ii) the initialisation scheme originally used by MOCK [15]; and (iii) the new initialisation scheme proposed in this paper. Due to the use of many method configurations and settings, this analysis focuses on our collection of 350 synthetic problems only. The results are summarised in Fig. 12.

By exploring different settings for parameter δ, it is possible to appreciate from Fig. 12 the extent to which different encoding lengths, and therefore sizes of the resulting search spaces, can affect clustering performance to varying degrees depending on the initialisation scheme under consideration.[19] It is interesting to observe that as better-quality genetic material is introduced in the initial population (through the use of a more effective initialisation scheme), less evident becomes the impact that parameter δ has on performance. The use of the random initialisation strategy, *i.e.* R^L_δ and R^B_δ in the figure, exhibits a clear trend of improvement as δ increases (except for the drop in performance observed when using $\delta = sr1$, see discussion below). The performance of these approaches goes from being notably poor when using low values for δ, to being considerably more competitive when using high values for δ. These stepped enhancements in clustering performance are the result of the increasingly significant reduction of the search space which is achieved in response to the rise in δ. Similar tendencies (though not as pronounced as for the random initialisation) can be seen for approaches L_δ and B_δ using the original initialisation of MOCK.

On the other hand, a different trend is shown by Δ-MOCK configurations Δ^L_δ and Δ^B_δ, which present the most remarkable overall results in Fig. 12. The steady behaviour of these approaches in most cases suggests that the setting of δ does not seem to (seriously) affect performance if using Δ-MOCK's initialisation. As expected, however, there is a point at which the excessive reduction in the encoding length causes an inevitable drop in performance (which is observed for all approaches analysed here). Setting $\delta = sr1$, in particular, results in encodings of length $|\Gamma| \approx \sqrt{N}$, which prevents the method from producing partitions with the correct number of clusters for many of the clustering problems considered in this study. For instance, Fig. 10 indicates that for problem configurations with $k^* = 120$ clusters, $|\Gamma| < 100$ in all the cases; thus, the reduced-length representations are clearly unable to encode partitions for which k is close to k^* under this setting.[20] In fact, for all the 150 problem instances with $k^* \in \{80, 100, 120\}$, the encoding length using $\delta = sr1$ is found to be $|\Gamma| < k^* - 1$, which explains the low ARI values scored using this setting. Nevertheless, these numbers of clusters $k^* \geq 80$ are particularly large and are not representative of most practical clustering problems (in spite of this, these large values of k^* are considered in this study with the purpose of evaluating Δ-MOCK in a wide range of clustering conditions). Moreover, there are many clustering applications for which k^* grows slowly (or even has a fixed small value), while N is considerably large and grows at a much faster rate. In such clustering scenarios, setting $\delta = sr1$ (as well as other settings which can achieve even further reductions in the length of the encoding) might still offer a competitive advantage.

The absence of perceptible differences in Δ-MOCK's clustering performance, in spite of the use of representations of considerably different lengths due to the changing value of δ, can be explained by two facts. First, the new initialisation routine of Δ-MOCK is significantly more effective and substantially narrows the gap between the starting performance (initial population) and the best performance that can be reached by the search method for a given data set (see Section V-B). In other words, this initialisation routine reduces the margin of potential improvement, which certainly renders the contributions of the reduced-length encodings less evident.

Second, the use of a specialised initialisation routine significantly biases exploration. Δ-MOCK's initialisation achieves a comparable (and, to a certain extent, redundant) effect to that of the reduced-length encodings, implicitly constraining the portion of the solution space which is effectively reached and exploited by the method. This is because in a method like Δ-MOCK (and MOCK), where the generation of new candidate individuals during optimisation mainly relies on recombination (given the low mutation rates used), the genetic material introduced during initialisation plays a key role in delimiting the region of the solution space where the efforts are concentrated. It also seems naturally appropriate to argue that the length of the encoding becomes less influential if, regardless of the total encoding length, only a small subset of the encoding positions become the primary target of the

[19]Note that, when $\delta = 0$, L_δ is equivalent to the baseline approach M^*_V.

[20]The maximum number of clusters that can be encoded is $|\Gamma| + 1$ (as in the partial solution defined by the set of fixed MST links Δ, see Fig. 2). Note, however, that the vast majority of the genotypes for the reduced-length representations encode partitions with $k < |\Gamma| + 1$, especially in the case of the Δ-locus encoding due to the link replacement strategy adopted.

optimisation due to the bias induced by the initialisation. This behaviour is accentuated in this scenario given that the same criterion (based on the ranking of the MST links) used to reduce the length of the encoding is exploited to guide initialisation. Hence, both Δ-MOCK's initialisation and representation schemes are able to independently provide, implicitly and explicitly, respectively, an advantageous reduction of the accessible solution space, and therefore an important boost in performance as evidenced by the experiments of this paper. The combined use of these strategies in Δ-MOCK, however, preserves these performance advantages while exploiting the benefits that these strategies individually entail in terms of computational efficiency (as evaluated in Section V-D).

VI. Discussion

Δ-MOCK presents significant improvements with respect to its predecessor MOCK. The change in the search strategy, replacing (the strongly elitist) PESA-II with (the less elitist) NSGA-II, has been found to provide a competitive advantage in this particular optimisation scenario, as it increases the method's ability to exploit the high-quality partitions generated during initialisation (Section V-A). Although this change in the underlying search engine is not seen as one of Δ-MOCK's main original contributions, it is seen as an important first step in the interest of capitalising on the benefits that the new initialisation scheme proposed in this paper can provide.

The new initialisation routine of Δ-MOCK was found to substantially heighten the quality of the initial populations, particularly when dealing with problems which involve challenging, elongated cluster shapes (Section V-B). Moreover, this specialised initialisation strategy introduces a strong bias and, implicitly, helps to overcome the difficulties related to the size of the search space in an effective manner (Section V-E). The new reduced-length representations of Δ-MOCK further contribute in this respect, explicitly pruning large portions of the search space so that exploration can focus on its most promising regions only (Section III-G). Consequently, the boost achieved in the starting performance and the (implicit and explicit) reductions of the search space allow Δ-MOCK to produce PFAs with notably better characteristics than those produced by MOCK (Section V-C). This was observed to reliably translate into an increased proficiency at finding high-quality clustering solutions (Section V-A).

Besides these meaningful advantages in terms of clustering performance, the new initialisation and representation schemes address the primary sources of computational overhead identified in the original implementation (Section V-D). On the one hand, the new initialisation routine removes the use of the k-means algorithm, which is a computationally intensive design component of MOCK's initialisation strategy. On the other hand, the new representations allow Δ-MOCK to operate on genotypes of a significantly reduced length, which are thus more efficient to handle. Furthermore, these representations enable delta-evaluation, which helps to alleviate the computational burden derived from the decoding and evaluation of candidate individuals during the search process. Overall, therefore, it can be said with certainty that both the new initialisation routine and the new reduced-length genetic representations are valuable contributors that account for Δ-MOCK's clustering and computational performance.

Focusing on the new genetic representations, our experiments have shown that it is actually possible to use encoding lengths that grow at a much slower (non-linear) rate with respect to the increase in the problem size N (Section V-E). More specifically, very encouraging results (as discussed above) were obtained in this study when using encoding lengths defined as a function of $\sqrt{N}$, which redefine the problem as one of clearly decreased complexity. Nevertheless, it was also found that excessive reductions in the length of the encoding can discard key regions of the search space and therefore prevent the discovery of good-quality, globally optimal clustering solutions (a risk that does not exist when using the full-length representation of MOCK). Our results also suggest, however, that such encoding length reductions (and even more significant reductions than the ones studied here), considered excessive for our collection of test data sets, might find their applicability in clustering scenarios with a considerably large N; and, particularly, in scenarios where N grows at a much faster rate in comparison to the number of clusters k. These characteristics are common to many practical clustering applications. In market (customer) segmentation [2], [3], for instance, there is usually a finite number of segments (clusters) but the density of customers (N) could increase at an extremely high rate (especially in a growing market) and only be constrained by factors such as, *e.g.*, population size. The investigation of our reduced-length representations (and of the Δ-MOCK algorithm in general) in the context of such type of practically relevant applications will constitute one of the main directions for our future work.

Evidently, a key factor around this topic is the effectiveness of the mechanism through which encoding length reduction is accomplished. The more effective this mechanism, the more we should be able to reduce the length of the encoding (and corresponding search space) without sacrificing performance (there exist, however, obvious restrictions: the encoding length imposes a limit in the maximum k that a partition may have under the new representations, and the encoding needs to allow the search method to explore a diversity of candidate partitions with a range of interesting values of k). In Δ-MOCK, we adopted a mechanism that relies on the ranking of the MST links on the basis of their degree of interestingness. Such a criterion has been found to be more effective than other criteria in discriminating between the MST links (results of such an assessment are included in our supplementary material). Due to the critical role that this strategy plays, a more thorough analysis of its effectiveness, as well as the design of new alternative strategies (not necessarily relying on the MST), is seen as a promising avenue for further research.

Both the two reduced-length representations of Δ-MOCK reported very competitive (and to some extent comparable) clustering performances. Nonetheless, our experiments indicate that the Δ-locus encoding tends to produce better results than the Δ-binary encoding. On the other hand, the Δ-binary encoding was found to reach a better performance than the Δ-locus encoding during initial stages of the search process

in many problems (in most cases being outperformed by the Δ-locus encoding at later stages though), and also scored the best results for some of our low-dimensional problems. These representations need to be further investigated in a wider range of clustering scenarios in order to draw more general conclusions. It seems especially necessary to identify and characterise the conditions under which one particular encoding scheme can be more advantageous than the other.

Finally, an essential research direction is that concerned with the process of model selection. This study was devoted to the development and investigation of our new algorithm Δ-MOCK which, in comparison to the original algorithm MOCK, is capable of producing higher-quality PFAs containing more promising candidate partitions (also achieving important savings in the computational costs as discusses above). These approximation sets not only comprise different trade-offs between Δ-MOCK's optimisation criteria, but the partitions may also present a wide diversity of k values (Section V-C, Fig. 8). This enables the exploration of the hierarchical structure of the data (where this exists) [15]. Furthermore, having the opportunity to analyse multiple alternative solutions facilitates the exploitation of any specialised domain expertise available, which can certainly boost the effectiveness of exploratory data analysis. However, the ultimate goal in many applications requires the (automated) selection of only one (or a small subset) of the candidate partitions generated as the problem solution. MOCK's approach to model selection is based on the assumption that the structure of the data is reflected in the shape of the PFA; in this way, MOCK exploits information about the shape of the PFAs obtained as a means to guide the selection of the most promising candidate partitions. Δ-MOCK's improvements to the characteristics of the PFAs, and the changes that this algorithm involves in the optimisation criteria, are not expected to compromise the effectiveness of the original model selection strategy. We ought to dedicate future research efforts to verify this and to identify potential opportunities for further development in this context.

A. Contributions

In summary, we see the key contribution of this work as follows. Firstly, we provide new insight regarding the heuristic bias intrinsic to MOCK's original encoding, and the implications that this has for the algorithm's scalability. This is of particular importance as MOCK's linkage-based encoding has been previously criticised in the literature, as it increases linearly with data set size, while a centroid-based encoding remains constant. Our experiments shed new light on this issue and show that the effective combination of our flexible encoding with a suitable seeding mechanism ensures a consistent advantage over a centroid-based encoding, even for large data sets. Secondly, we use the insight regarding MOCK's search space to propose two reduced-length encodings and provide an analysis of the impact of encoding length on search performance, and how this links to problem difficulty. Finally, we revisit the key modelling choices in MOCK's cluster generation phase, and introduce an improved algorithm, Δ-MOCK, that shows a highly significant decrease in computational expense and improvements in clustering accuracy for the majority of the data sets considered.

VII. Conclusion

Clustering is a fundamental tool in exploratory data analysis, but the unprecedented volumes of data generated nowadays can represent a major bottleneck and compromise the effectiveness and limit the applicability of current techniques. In light of this, we described and studied a new clustering algorithm in this paper: Δ-MOCK. Our Δ-MOCK algorithm implements an evolutionary multiobjective approach to the problem of data clustering, and seeks to improve upon the overall scalability of its predecessor algorithm MOCK [15].

The most significant adaptations introduced by Δ-MOCK are the redesign of its specialised initialisation routine and the use of novel compact representations (additional adaptations include changes to the underlying search engine and optimisation criteria). In this study, we have investigated, analysed, and discussed the relevance of each individual adaptation. In general, these changes exert a strong influence in the size of the search space that is effectively reached by the method and enable a more computationally efficient generation and processing of candidate clustering solutions. A rigorous empirical evaluation has shown that Δ-MOCK outperformed MOCK across a diverse set of clustering scenarios, achieving significant increases in both performance and efficiency. These findings confirm that our new algorithm can provide meaningful advantages in practice, and should lead to a tangible improvement in our ability to address the larger and more challenging clustering problems that the 'big data' era poses.

Appendix A
Tables of Results

This appendix includes tables complementing the results of the experiments presented in Section V. Regarding clustering performance, Table III details the results of the ARI measure for the different methods studied in this paper. In terms of the PFAs obtained, Table IV, presents the results for the HV and IGD^+ performance indicators.

Acknowledgment

This work is funded by grant EP/M013766/1, Engineering and Physical Sciences Research Council, UK. The first author acknowledges complementary support through a postdoctoral fellowship from CONACYT-Mexico.

References

[1] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, Jun 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167865509002323

[2] J.-J. Huang, G.-H. Tzeng, and C.-S. Ong, "Marketing segmentation using support vector clustering," *Expert Systems with Applications*, vol. 32, no. 2, pp. 313–317, Feb 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0957417405003404

[3] K. Tsiptsis and A. Chorianopoulos, *Data Mining Techniques in CRM: Inside Customer Segmentation*. John Wiley & Sons, Ltd, Jan 2010.

[4] Y. Xu, V. Olman, and D. Xu, "Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees," *Bioinformatics*, vol. 18, no. 4, pp. 536–545, Apr 2002. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/18.4.536

TABLE III

DETAILED RESULTS FOR THE ARI MEASURE ON ALL UKC DATA SETS AND SYNTHETIC PROBLEM CONFIGURATIONS. THE PERFORMANCE OF Δ-MOCK APPROACHES Δ^L_{sr5} AND Δ^B_{sr5} IS COMPARED WITH RESPECT TO THE BASELINE MOCK CONFIGURATION M^*_V. RESULTS FOR THE ORIGINAL MOCK (M^{H07}) AND REFERENCE METHODS k-MEANS, MGA, MGA_{VL}, AND MGA^*_{VL} ARE ALSO INCLUDED AS A REFERENCE. IN ALL THE CASES, THE AVERAGE k OF THE SOLUTIONS SELECTED DURING THE MEASURE COMPUTATION (SEE SECTION IV-D) IS INDICATED IN PARENTHESIS. THE BEST (HIGHEST) ARI SCORED FOR EVERY PROBLEM HAS BEEN SHADED AND HIGHLIGHTED IN BOLD. THE FINDINGS OF THE STATISTICAL SIGNIFICANCE ANALYSIS ARE REPORTED AS FOLLOWS. RESULTS FOR Δ^L_{sr5} AND Δ^B_{sr5} ARE MARKED • IF THERE IS A STATISTICALLY SIGNIFICANT DIFFERENCE WITH RESPECT TO M^*_V. IN ADDITION, RESULTS OF Δ^B_{sr5} ARE MARKED ⋆ IF THERE IS A STATISTICALLY SIGNIFICANT DIFFERENCE WITH RESPECT TO Δ^L_{sr5}.

Problem	N	k-means	MGA	MGA_{VL}	MGA^*_{VL}	M^{H07}	M^*_V	Δ^L_{sr5}	Δ^B_{sr5}
UKC1	29463	0.991 (11.0)	0.994 (11.0)	0.915 (16.0)	0.992 (13.5)	**0.999 (11.3)**	0.997 (12.0)	0.996 (13.4) •	0.996 (13.0) •
UKC2	26739	0.969 (10.0)	0.969 (10.0)	0.926 (15.1)	**0.985 (11.3)**	0.978 (11.0)	0.977 (12.0)	0.977 (12.0)	0.977 (12.0)
UKC3	31929	0.961 (11.9)	0.890 (11.0)	0.948 (13.4)	0.968 (13.9)	0.998 (11.2)	**0.999 (11.9)**	0.992 (16.0) •	0.993 (14.0) •
UKC4	29149	0.997 (10.0)	**0.998 (10.0)**	**0.998 (10.4)**	0.976 (12.5)	**0.998 (11.0)**	0.997 (11.9)	0.989 (15.0) •	0.991 (14.2) •
UKC5	30688	0.935 (11.3)	0.941 (11.0)	0.835 (16.9)	0.900 (16.4)	0.985 (12.0)	**0.990 (12.5)**	0.989 (13.9) •	**0.990 (13.0)** •
UKC6	31191	0.911 (12.0)	0.842 (11.0)	0.854 (17.4)	0.925 (12.2)	0.936 (14.4)	**0.960 (14.0)**	0.948 (15.7)	0.945 (15.1) •
UKC7	31666	0.889 (16.0)	0.931 (11.0)	0.868 (15.3)	0.911 (14.2)	0.971 (12.6)	0.976 (13.6)	**0.987 (15.0)** •	0.983 (14.7)
UKC8	34654	0.846 (12.6)	0.849 (12.0)	0.809 (13.9)	0.838 (14.1)	0.942 (16.7)	**0.950 (16.4)**	0.932 (18.1) •	0.933 (17.4) •
20d-10c	2768	0.508 (17.4)	0.440 (10.0)	0.500 (18.3)	0.585 (16.8)	0.996 (10.1)	0.998 (10.0)	**1.000 (10.0)** •	**1.000 (10.0)** •
50d-10c	2747	0.470 (18.7)	0.419 (10.0)	0.469 (18.1)	0.617 (15.9)	0.998 (10.0)	0.999 (10.0)	**1.000 (10.0)** •	**1.000 (10.0)** •
100d-10c	2664	0.495 (16.9)	0.443 (10.0)	0.491 (18.5)	0.670 (14.6)	0.998 (10.1)	0.999 (10.0)	**1.000 (10.0)** •	**1.000 (10.0)** •
150d-10c	2859	0.485 (18.0)	0.448 (10.0)	0.479 (18.4)	0.667 (13.7)	0.998 (10.0)	0.999 (10.0)	**1.000 (10.0)** •	**1.000 (10.0)** •
200d-10c	2724	0.508 (17.0)	0.454 (10.0)	0.509 (18.5)	0.715 (13.6)	0.999 (10.0)	**1.000 (10.0)**	**1.000 (10.0)** •	**1.000 (10.0)** •
20d-20c	5587	0.515 (35.6)	0.382 (20.0)	0.485 (39.8)	0.538 (37.3)	0.983 (19.9)	0.993 (20.1)	**0.997 (20.1)** •	0.996 (20.1) •
50d-20c	5398	0.490 (39.4)	0.362 (20.0)	0.467 (39.7)	0.534 (37.3)	0.990 (19.9)	0.996 (19.9)	**1.000 (20.0)** •	**1.000 (20.0)** •
100d-20c	5393	0.500 (38.5)	0.365 (20.0)	0.479 (39.3)	0.548 (36.7)	0.993 (20.0)	0.997 (20.0)	**1.000 (20.0)** •	**1.000 (20.0)** •
150d-20c	5681	0.489 (39.8)	0.361 (20.0)	0.471 (39.4)	0.565 (34.2)	0.993 (19.8)	0.997 (19.9)	**1.000 (20.0)** •	**1.000 (20.0)** •
200d-20c	5327	0.498 (40.3)	0.363 (20.0)	0.483 (39.4)	0.578 (34.4)	0.997 (20.0)	0.999 (20.0)	**1.000 (20.0)** •	**1.000 (20.0)** •
20d-40c	2982	0.545 (77.1)	0.285 (40.0)	0.502 (79.7)	0.533 (76.5)	0.806 (50.0)	0.858 (47.9)	0.928 (43.8) •	**0.932 (44.7)** •
50d-40c	3033	0.540 (85.8)	0.253 (40.0)	0.519 (79.9)	0.547 (78.4)	0.888 (45.2)	0.927 (43.9)	**0.975 (42.0)** •	0.974 (42.9) •
100d-40c	3030	0.533 (87.1)	0.257 (40.0)	0.519 (79.9)	0.544 (78.3)	0.892 (45.0)	0.931 (43.0)	**0.987 (41.2)** •	0.983 (41.9) • ⋆
150d-40c	3025	0.527 (90.0)	0.250 (40.0)	0.514 (79.9)	0.549 (78.0)	0.885 (45.0)	0.928 (43.2)	**0.986 (41.3)** •	0.983 (42.2) • ⋆
200d-40c	3022	0.515 (89.7)	0.246 (40.0)	0.504 (79.8)	0.532 (78.1)	0.898 (43.8)	0.940 (42.5)	**0.989 (41.0)** •	0.986 (41.8) • ⋆
20d-60c	4476	0.539 (103.1)	0.258 (60.0)	0.510 (119.4)	0.524 (115.8)	0.781 (78.8)	0.833 (76.0)	**0.907 (66.9)** •	**0.907 (69.7)** •
50d-60c	4546	0.508 (103.4)	0.208 (60.0)	0.525 (119.7)	0.537 (118.3)	0.874 (69.4)	0.912 (68.0)	**0.968 (64.3)** •	0.964 (65.8) •
100d-60c	4529	0.472 (101.0)	0.204 (60.0)	0.521 (119.9)	0.533 (118.5)	0.867 (70.1)	0.913 (67.6)	**0.976 (63.2)** •	0.972 (64.2) • ⋆
150d-60c	4513	0.459 (101.5)	0.198 (60.0)	0.519 (119.7)	0.535 (118.3)	0.865 (70.2)	0.916 (67.1)	**0.980 (62.8)** •	0.975 (64.4) • ⋆
200d-60c	4526	0.434 (102.3)	0.189 (60.0)	0.502 (119.9)	0.514 (118.5)	0.866 (69.2)	0.909 (66.6)	**0.982 (62.8)** •	0.978 (64.1) • ⋆
20d-80c	5950	0.520 (124.5)	0.252 (80.0)	0.505 (158.9)	0.515 (155.7)	0.751 (111.5)	0.805 (109.1)	0.884 (91.8) •	**0.887 (95.9)** •
50d-80c	6054	0.472 (122.4)	0.193 (80.0)	0.522 (159.8)	0.533 (157.2)	0.847 (97.4)	0.889 (95.8)	**0.960 (87.4)** •	0.956 (89.4) • ⋆
100d-80c	6049	0.416 (121.2)	0.170 (80.0)	0.518 (159.8)	0.525 (158.1)	0.851 (95.9)	0.900 (92.8)	**0.969 (86.3)** •	0.966 (88.2) • ⋆
150d-80c	6043	0.396 (120.9)	0.165 (80.0)	0.512 (159.4)	0.515 (158.3)	0.861 (92.9)	0.904 (91.3)	**0.973 (85.5)** •	0.969 (87.1) • ⋆
200d-80c	6018	0.368 (121.1)	0.158 (80.0)	0.502 (159.8)	0.507 (158.4)	0.817 (96.7)	0.881 (93.9)	**0.973 (85.5)** •	0.969 (87.5) • ⋆
20d-100c	7465	0.512 (144.9)	0.262 (100.0)	0.509 (198.9)	0.517 (193.9)	0.743 (140.9)	0.794 (139.9)	0.852 (118.0) •	**0.855 (122.3)** •
50d-100c	7554	0.446 (142.7)	0.182 (100.0)	0.522 (199.8)	0.525 (197.3)	0.822 (126.0)	0.867 (124.9)	**0.943 (110.3)** •	**0.943 (114.1)** •
100d-100c	7532	0.367 (141.0)	0.156 (100.0)	0.513 (199.8)	0.518 (198.2)	0.825 (123.7)	0.881 (119.3)	**0.961 (109.2)** •	0.957 (111.6) • ⋆
150d-100c	7525	0.348 (141.5)	0.149 (100.0)	0.518 (199.7)	0.515 (198.3)	0.834 (119.7)	0.890 (117.4)	**0.970 (108.0)** •	0.965 (110.6) • ⋆
200d-100c	7460	0.315 (139.8)	0.141 (100.0)	0.500 (199.8)	0.501 (198.2)	0.804 (122.8)	0.876 (119.3)	**0.967 (108.8)** •	0.962 (111.0) • ⋆
20d-120c	8960	0.497 (164.5)	0.263 (120.0)	0.507 (238.7)	0.515 (235.0)	0.722 (175.0)	0.771 (178.5)	0.843 (141.3) •	**0.851 (148.6)** • ⋆
50d-120c	9116	0.419 (163.3)	0.174 (120.0)	0.521 (239.4)	0.519 (237.2)	0.815 (154.4)	0.860 (152.1)	**0.940 (134.3)** •	0.937 (137.8) •
100d-120c	9068	0.334 (161.6)	0.151 (120.0)	0.511 (239.8)	0.515 (237.8)	0.806 (150.9)	0.864 (147.0)	**0.956 (131.8)** •	0.950 (135.4) • ⋆
150d-120c	9028	0.301 (160.7)	0.138 (120.0)	0.508 (239.6)	0.512 (237.9)	0.799 (151.2)	0.873 (146.9)	**0.960 (131.7)** •	0.956 (135.1) • ⋆
200d-120c	8939	0.272 (161.5)	0.132 (120.0)	0.500 (239.7)	0.493 (238.3)	0.794 (149.1)	0.868 (144.0)	**0.961 (131.7)** •	0.956 (134.5) • ⋆

[5] R. S. H. Istepanian, A. Sungoor, and J.-C. Nebel, "Comparative analysis of genomic signal processing for microarray data clustering," *IEEE Transactions on NanoBioscience*, vol. 10, no. 4, pp. 225–238, Dec 2011. [Online]. Available: http://ieeexplore.ieee.org/document/6096429/

[6] U. Maulik, S. Bandyopadhyay, and A. Mukhopadhyay, *Multiobjective Genetic Algorithms for Clustering*. Springer Berlin Heidelberg, 2011.

[7] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falco, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 50–68, Jun 2009. [Online]. Available: http://doi.wiley.com/10.1002/ima.20191

[8] Z. Ma, J. M. R. Tavares, R. N. Jorge, and T. Mascarenhas, "A review of algorithms for medical image segmentation and their applications to the female pelvic cavity," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 13, no. 2, pp. 235–246, Apr 2010. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10255840903131878

[9] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surveys*, vol. 31, no. 3, pp. 264–323, Sep 1999.

[10] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics, Jan 2007.

[11] J. M. Kleinberg, "An Impossibility Theorem for Clustering," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, 2003, pp. 463–470.

[12] J. Handl and J. Knowles, "Evolutionary Multiobjective Clustering," in *Parallel Problem Solving from Nature*, X. Y. et al., Ed. Birmingham, UK: Springer Berlin Heidelberg, September 2004, pp. 1081–1091.

[13] ——, "Multiobjective Clustering with Automatic Determination of the Number of Clusters," UMIST, Manchester, UK, Tech. Rep. TR-COMPSYSBIO-2004-02, August 2004.

[14] ——, *Exploiting the Trade-off – The Benefits of Multiple Objectives in Data Clustering*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, vol. 3410, ch. chapter 38, pp. 547–560. [Online]. Available: http://link.springer.com/10.1007/978-3-540-31880-4_38

[15] ——, "An evolutionary approach to multiobjective clustering," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 56–76, Feb 2007. [Online]. Available: http://ieeexplore.ieee.org/document/

TABLE IV
RESULTS FOR THE HV AND IGD^+ PERFORMANCE INDICATORS ON ALL UKC DATA SETS AND SYNTHETIC PROBLEMS. PERFORMANCE OF Δ-MOCK APPROACHES Δ^L_{sr5} AND Δ^B_{sr5} IS COMPARED WITH RESPECT TO THE BASELINE MOCK CONFIGURATION M^*_V. FOR ALL PROBLEMS, THE BEST RESULT SCORED FOR EVERY PERFORMANCE INDICATOR HAS BEEN SHADED AND HIGHLIGHTED IN BOLD (WHEREAS HV IS TO BE MAXIMISED, IGD^+ IS TO BE MINIMISED). VALUES FOR IGD^+ HAVE BEEN MULTIPLIED BY 10^2 IN ALL THE CASES. RESULTS OF THE STATISTICAL SIGNIFICANCE ANALYSIS ARE SHOWN AS DESCRIBED IN TABLE III.

	HV			$IGD^+ \times 10^2$		
Problem	M^*_V	Δ^L_{sr5}	Δ^B_{sr5}	M^*_V	Δ^L_{sr5}	Δ^B_{sr5}
UKC1	0.850	**0.881** •	0.879 •	8.101	**2.555** •	3.755 • ⋆
UKC2	0.840	**0.855** •	0.851	6.645	**2.265** •	3.090 • ⋆
UKC3	0.954	**0.970** •	0.968 •	4.339	**1.457** •	2.324 • ⋆
UKC4	0.784	0.821 •	**0.825** •	7.316	**2.759** •	3.337 •
UKC5	**0.974**	0.958 •	0.963 •	**1.871**	2.224	2.174
UKC6	0.977	0.979	**0.983** •	3.582	**1.708** •	1.839 •
UKC7	0.923	0.957 •	**0.959** •	5.295	**1.254** •	1.718 • ⋆
UKC8	0.977	0.975	**0.978**	1.770	**1.170** •	1.363 •
20d-10c	0.932	**0.940** •	**0.940** •	3.011	**1.540** •	1.985 • ⋆
50d-10c	0.887	0.900 •	**0.902** •	5.351	**2.512** •	2.650 •
100d-10c	0.872	0.888 •	**0.893** •	5.972	**2.415** •	2.559 •
150d-10c	0.866	0.879 •	**0.882** •	5.136	**2.616** •	2.812 •
200d-10c	0.841	0.858 •	**0.863** •	6.028	2.688 •	**2.661** •
20d-20c	0.960	0.966 •	**0.968** •	2.680	**1.294** •	1.598 • ⋆
50d-20c	0.934	0.945 •	**0.948** •	3.840	**1.427** •	1.645 • ⋆
100d-20c	0.861	0.883 •	**0.887** •	5.928	**2.058** •	2.433 • ⋆
150d-20c	0.839	0.859 •	**0.864** •	6.603	**2.501** •	2.722 •
200d-20c	0.815	0.838 •	**0.845** •	6.725	2.651 •	**2.638** •
20d-40c	0.916	0.917	**0.925** • ⋆	6.437	**2.426** •	2.650 • ⋆
50d-40c	0.936	0.946 •	**0.950** • ⋆	6.556	**1.757** •	2.124 • ⋆
100d-40c	0.940	0.954 •	**0.958** • ⋆	7.867	**1.551** •	2.021 • ⋆
150d-40c	0.943	0.958 •	**0.963** • ⋆	8.208	**1.596** •	1.999 • ⋆
200d-40c	0.943	0.959 •	**0.963** • ⋆	7.676	**1.442** •	1.801 • ⋆
20d-60c	0.899	0.897	**0.908** • ⋆	8.817	**2.803** •	3.508 • ⋆
50d-60c	0.931	0.941 •	**0.947** • ⋆	9.854	**2.285** •	2.924 • ⋆
100d-60c	0.936	0.950 •	**0.956** • ⋆	11.589	**2.063** •	2.673 • ⋆
150d-60c	0.935	0.955 •	**0.960** • ⋆	11.628	**2.022** •	2.718 • ⋆
200d-60c	0.935	0.957 •	**0.963** • ⋆	13.306	**1.872** •	2.554 • ⋆
20d-80c	0.884	0.886	**0.899** • ⋆	10.528	**3.113** •	3.705 • ⋆
50d-80c	0.925	0.939 •	**0.948** • ⋆	12.145	**2.503** •	3.173 • ⋆
100d-80c	0.930	0.948 •	**0.957** • ⋆	14.118	**2.343** •	2.973 • ⋆
150d-80c	0.930	0.952 •	**0.959** • ⋆	13.050	**2.200** •	2.856 • ⋆
200d-80c	0.926	0.957 •	**0.964** • ⋆	16.849	**2.215** •	2.885 • ⋆
20d-100c	0.874	0.875	**0.890** • ⋆	11.946	**3.487** •	4.128 • ⋆
50d-100c	0.913	0.929 •	**0.940** • ⋆	15.121	**3.065** •	3.541 • ⋆
100d-100c	0.922	0.944 •	**0.954** • ⋆	16.698	**2.728** •	3.244 • ⋆
150d-100c	0.926	0.951 •	**0.960** • ⋆	14.880	**2.418** •	2.949 • ⋆
200d-100c	0.928	0.956 •	**0.965** • ⋆	16.647	**2.344** •	2.897 • ⋆
20d-120c	0.866	0.868	**0.884** • ⋆	13.994	**3.866** •	4.304 • ⋆
50d-120c	0.910	0.928 •	**0.941** • ⋆	15.994	**3.235** •	3.726 • ⋆
100d-120c	0.917	0.940 •	**0.952** • ⋆	18.009	**3.042** •	3.614 • ⋆
150d-120c	0.921	0.947 •	**0.957** • ⋆	16.749	**2.781** •	3.149 • ⋆
200d-120c	0.922	0.950 •	**0.961** • ⋆	17.148	**2.676** •	3.085 • ⋆

4079614/

[16] M. Delattre and P. Hansen, "Bicriterion cluster analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 4, pp. 277–291, Jul 1980. [Online]. Available: http://ieeexplore.ieee.org/document/4767027/

[17] J. Handl and J. Knowles, "Improvements to the Scalability of Multiobjective Clustering," in *IEEE Congress on Evolutionary Computation*, vol. 3, 2005, pp. 2372–2379.

[18] ——, "Multiobjective Clustering Around Medoids," in *IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 632–639.

[19] A. Garcia-Piquer, J. Bacardit, A. Fornells, and E. Golobardes, "Scaling-up multiobjective evolutionary clustering algorithms using stratification," *Pattern Recognition Letters*, 2016, in Press. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865516303488

[20] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, CA, USA: University of California Press, 1967, pp. 281–297.

[21] S. Bandyopadhyay, U. Maulik, and A. Mukhopadhyay, "Multiobjective genetic clustering for pixel classification in remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 5, pp. 1506–1511, May 2007.

[22] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *J. Intell. Inform. Syst.*, vol. 17, no. 2/3, pp. 107–145, 2001.

[23] A. Ferligoj and V. Batagelj, "Direct multicriteria clustering algorithms," *Journal of Classification*, vol. 9, no. 1, pp. 43–61, Jan 1992. [Online]. Available: http://link.springer.com/10.1007/BF02618467

[24] V. Pareto, *Cours d'Economie Politique*. Genève: Droz, 1896.

[25] E. Hruschka, R. Campello, A. Freitas, and A. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*, vol. 39, no. 2, pp. 133–155, Mar 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4783080/

[26] S. Rana, S. Jasola, and R. Kumar, "A review on particle swarm optimization algorithms and their applications to data clustering," *Artificial Intelligence Review*, vol. 35, no. 3, pp. 211–222, Mar 2011. [Online]. Available: http://link.springer.com/10.1007/s10462-010-9191-9

[27] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary Computation*, vol. 16, pp. 1–18, Jun 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S221065021300076X

[28] A. A. A. Esmin, R. A. Coelho, and S. Matwin, "A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data," *Artificial Intelligence Review*, vol. 44, no. 1, pp. 23–45, Jun 2015. [Online]. Available: http://link.springer.com/10.1007/s10462-013-9400-4

[29] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part ii," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 20–35, Feb 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6658840/

[30] A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay, "A survey of multiobjective evolutionary clustering," *ACM Computing Surveys*, vol. 47, no. 4, pp. 1–46, May 2015. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2775083.2742642

[31] J. Handl and J. Knowles, "An Investigation of Representations and Operators for Evolutionary Data Clustering with a Variable Number of Clusters," in *Parallel Problem Solving from Nature*. Springer, 2006, pp. 839–849.

[32] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *J. Heuristics*, vol. 2, pp. 5–30, 1996.

[33] F. Rothlauf and D. E. Goldberg, "Redundant representations in evolutionary computation," *Evol. Comput.*, vol. 11, no. 4, pp. 381–415, 2003.

[34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[35] Y. J. Park and M. S. Song, "A Genetic Algorithm for Clustering Problems," in *Genetic Programming*. Madison, WI, USA: Morgan Kaufmann, July 1998, pp. 568–575.

[36] G. Raidl and J. Gottlieb, "Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem," *Evol. Comput.*, vol. 13, no. 4, pp. 441–475, 2005.

[37] M. Garza-Fabre, J. Handl, and J. Knowles, "A New Reduced-Length Genetic Representation for Evolutionary Multiobjective Clustering," in *Evolutionary Multi-Criterion Optimization: 9th International Conference, EMO 2017*. Münster, Germany: Springer International Publishing, 2017, pp. 236–251.

[38] A. Casillas, M. T. G. de Lena, and R. Martnez, *Document Clustering into an Unknown Number of Clusters Using a Genetic Algorithm*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2807, ch. chapter 7, pp. 43–49. [Online]. Available: http://link.springer.com/10.1007/978-3-540-39398-6_7

[39] R. B. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Psychometrika*, vol. 3, pp. 1–27, 1974.

[40] J. Gower and G. Ross, "Minimum Spanning Trees and Single Linkage Cluster Analysis," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 18, no. 1, pp. 54–64, 1969.

[41] M. Laszlo and S. Mukherjee, "Minimum spanning tree partitioning algorithm for microaggregation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 7, pp. 902–911, Jul 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1432700/

[42] C. Zhong, D. Miao, and R. Wang, "A graph-theoretical clustering method based on two rounds of minimum spanning trees," *Pattern*

Recognition, vol. 43, no. 3, pp. 752–766, Mar 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0031320309002945

[43] Y. Zhou, O. Grygorash, and T. F. Hain, "Clustering with minimum spanning trees," *International Journal on Artificial Intelligence Tools*, vol. 20, no. 01, pp. 139–177, Feb 2011. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218213011000061

[44] C. Zahn, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Transactions on Computers*, vol. C-20, no. 1, pp. 68–86, 1971.

[45] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "Pesa-ii: Region-based selection in evolutionary multiobjective optimization," 2001, pp. 283–290.

[46] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr 2002.

[47] T. Chan, G. Golub, and R. LeVeque, "Algorithms for Computing the Sample Variance: Analysis and Recommendations," *The American Statistician*, vol. 37, no. 3, pp. 242–247, 1983.

[48] G. Syswerda, "Uniform Crossover in Genetic Algorithms," in *International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 2–9.

[49] S. Bandyopadhyay, A. Mukhopadhyay, and U. Maulik, "An improved algorithm for clustering gene expression data," *Bioinformatics*, vol. 23, no. 21, p. 2859, 2007. [Online]. Available: +http://dx.doi.org/10.1093/bioinformatics/btm418

[50] U. Maulik, A. Mukhopadhyay, and S. Bandyopadhyay, "Combining pareto-optimal clusters using supervised learning for identifying co-expressed genes," *BMC Bioinformatics*, vol. 10, no. 1, p. 27, 2009. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-10-27

[51] A. Mukhopadhyay and U. Maulik, "Unsupervised pixel classification in satellite imagery using multiobjective fuzzy clustering combined with svm classifier," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 4, pp. 1132–1138, April 2009.

[52] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 8, pp. 841–847, Aug 1991.

[53] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[54] A. Mukhopadhyay, S. Bandyopadhyay, and U. Maulik, "Analysis of microarray data using multiobjective variable string length genetic fuzzy clustering," in *2009 IEEE Congress on Evolutionary Computation*, May 2009, pp. 1313–1319.

[55] A. Mukhopadhyay and U. Maulik, "A multiobjective approach to mr brain image segmentation," *Applied Soft Computing*, vol. 11, no. 1, pp. 872–880, 2011.

[56] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *J. Amer. Stat. Assoc.*, vol. 66, pp. 846–850, 1971.

[57] V. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall, *Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, vol. 1993, ch. chapter 15, pp. 213–225. [Online]. Available: http://link.springer.com/10.1007/3-540-44719-9_15

[58] M. López-Ibáñez, L. Paquete, and T. Stützle, "Exploratory Analysis of Stochastic Local Search Algorithms in Biobjective Optimization," in *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, 2010, pp. 209–222.

[59] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov 1999.

[60] E. Zitzler, M. Laumanns, L. Thiele, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, 2003.

[61] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified Distance Calculation in Generational Distance and Inverted Generational Distance," in *Evolutionary Multi-Criterion Optimization*. Guimarães, Portugal: Springer International Publishing, 2015, pp. 110–125.

Mario Garza-Fabre Mario Garza-Fabre received the M.Sc. and Ph.D. degrees in Computer Science from the Center for Research and Advanced Studies of the National Polytechnic Institute (CINVESTAV), Mexico, in 2009 and 2014, respectively. He is now a Research Associate at the Decision and Cognitive Sciences Research Centre, University of Manchester, UK. His main research interests involve the analysis and design of heuristic optimisation techniques and their application to problems from areas such as bioinformatics and data mining/machine learning.

Julia Handl Julia Handl obtained a Bsc (Hons) in Computer Science from Monash University in 2001, an MSc degree in Computer Science from the University of Erlangen-Nuremberg in 2003, and a PhD in Bioinformatics from the University of Manchester in 2006. From 2007 to 2011, she held an MRC Special Training Fellowship at the University of Manchester, and she is now a Senior Lecturer (Associate Professor) in the Decision and Cognitive Sciences Group at the Alliance Manchester Business School. Her research includes theoretical and empirical work related to the development and use of data-mining and optimization approaches in a variety of application areas.

Joshua Knowles Joshua Knowles is Professor of Natural Computation at the School of Computer Science, University of Birmingham UK. He is associated with the development of early elitist EMO algorithms, PAES, PESA, and PESA-II, as well as grid-based archiving, proofs of convergence and diversity in EMO, No Free Lunch theorems for EMO, the duality/decomposition approach 'multiobjectivization' including theoretical work using drift arguments, performance assessment techniques and tools for EMO, the first algorithm using hypervolume for selection, MOCK multiobjective clustering, and the Bayesian optimization method, ParEGO. He has two children.