Towards an Adaptive Encoding for Evolutionary Data Clustering

DOI:
10.1145/3205455.3205506

Document Version
Accepted author manuscript

Published in:
GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference

Towards an Adaptive Encoding for Evolutionary Data Clustering

Cameron Shand
University of Manchester, UK
cameron.shand@manchester.ac.uk

Richard Allmendinger
University of Manchester, UK

Julia Handl
University of Manchester, UK

John Keane
University of Manchester, UK

ABSTRACT

A key consideration in developing optimization approaches for data clustering is choice of a suitable encoding. Existing encodings strike different trade-offs between model and search complexity, limiting the applicability to data sets with particular properties or to problems of moderate size. Recent research has introduced an additional hyperparameter to directly govern the encoding granularity in the multi-objective clustering algorithm MOCK. Here, we investigate adapting this important hyperparameter during run-time. In particular, we consider a number of different trigger mechanisms to control the timing of changes to this hyperparameter and strategies to rapidly explore the newly "opened" search space resulting from this change. Experimental results illustrate distinct performance differences between the approaches tested, which can be explained in light of the relative importance of initialization, crossover and mutation in MOCK. The most successful strategies meet the clustering performance achieved for an optimal (a priori) setting of the hyperparameter, at a ∼40% reduction of computational expense.

CCS CONCEPTS

• **Information systems → Clustering**; • **Theory of computation → Evolutionary algorithms**;

KEYWORDS

Evolutionary Multiobjective Clustering, Parameter Control

ACM Reference Format:
Cameron Shand, Richard Allmendinger, Julia Handl, and John Keane. 2018. Towards an Adaptive Encoding for Evolutionary Data Clustering. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Clustering aims to partition data into disjoint subsets/clusters where objects within a subset are 'similar' to each other, and dissimilar to objects in other clusters. Due to the subjective nature of what defines a cluster, a wide array of algorithms have been proposed which embrace different mathematical definitions. In particular, existing methods often incorporate specific model assumptions through the choice of clustering criterion or the choice of cluster representation — both of these aspects have important consequences regarding the types of clusters that can ultimately be identified.

The impossibility of defining a single universal clustering criterion has led to the formulation of clustering as a multi-objective problem involving optimization of a number of complementary criteria [4, 8]. Less attention has been paid to the choice of cluster representation which, arguably, acts as a more implicit mechanism of defining model structure, directly influencing search resolution. In the cognate literature, a number of different representations have been proposed, yet discussion of the implications of this choice has often been quite limited. Furthermore, a few authors have described evolutionary clustering (EC) algorithms with some level of control over the resolution of the search, but without much exploration of the impact and optimal determination of this parameter.

For example, in certain EC algorithms, the level of resolution is specified a priori in the form of the number of prototypes [9] (or Voronoi cells [7]) to be used during the search, with the assumption that multiple such prototypes (or cells) can subsequently be merged into a single cluster. The number of basic components defined in this manner can be thought of as an additional hyperparameter to the algorithm, which determines the resolution of the clustering (although the identity of each component may vary over the course of the search). The recently introduced multi-objective clustering algorithm Δ-MOCK [6] uses a related approach. Specifically, it derives a set of basic components using the information embedded in the Minimum Spanning Tree (MST) of a dataset of N items, with a hyperparameter controlling the number of components P to be defined. The resulting components are deterministic and are retained throughout the course of the search. This approach makes explicit that the search problem is transformed into one requiring optimal assembly of a number of components $P \ll N$ only.

A challenge to all of the approaches is the determination of the optimal choice of the resolution, or the number of components P. Intuitively, it seems clear that the optimal value of such a hyperparameter is likely to differ for datasets of varying complexity, as influenced by the shape and the number of clusters. Additionally, a particular application context may potentially add requirements regarding the optimal choice of the parameter. A suitably low resolution has the benefit of reducing the problem size and opens up the possibility of pre-computing objective value contributions for individual components. In certain applications where a single partition is desired, a low resolution that allows thorough exploration of the region around the true number of clusters (k^*) may yield clusters of sufficient accuracy. In other cases, however, k^* may be entirely unknown, or the nature of the application may warrant the generation of a broad range of diverse partitions. In such a scenario, the choice of a fixed low resolution may prove overly restrictive.

This paper investigates using feedback of the performance during run-time to change the resolution of the clustering model in an EC algorithm. Specifically, we devise strategies to achieve an incremental change from a low to a high resolution over the course

Conference'17, July 2017, Washington, DC, USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

of the search. We explore the impact of the timing of resolution changes, and describe a number of different search strategies for reacting to this change in resolution. The ultimate aim is to devise an algorithm that adaptively adjusts the resolution of the search, leading to improvements in efficiency and making it applicable in situations where optimal resolution for a given dataset is unknown.

The rest of this paper is organised as follows: Section 2 discusses related work; Section 3 introduces the Δ-MOCK algorithm, which serves as the vehicle for our experiments; Section 4 describes the trigger mechanisms and search strategies incorporated into our new version of the algorithm, Adaptive Δ-MOCK; Section 5 presents an experiment to compare various configurations of Adaptive Δ-MOCK on a range of both synthetic and real-world datasets, followed by analyses of the results; Section 6 presents our conclusions.

2 RELATED WORK

Performance of an optimization algorithm greatly depends on the values of its parameters. Setting these parameters efficiently can depend not only on the problem at hand but also the purpose of optimization, such as the need for either a single or set of solutions, or for a solution that is 'good enough' within a time constraint. The No Free Lunch Theorem [19] states there is no single optimal algorithm or setting that yields optimal performance on all problem classes.

Much recent research has focused on appropriate setting of algorithm parameters, see e.g. [10]. It is common to differentiate the following two cases [5]: parameter tuning and parameter control. In parameter tuning (or offline tuning), the parameter values are unchanged during optimization, hence only a single value is needed for each parameter. In parameter control (or online tuning), the parameter values are changed during an optimization run. In particular, starting off with initial parameter values, the values may be modified during a run via suitable control strategies, which in turn can be deterministic, adaptive (using feedback from the search), or self-adaptive (the values are included directly in the search).

The approach here, Adaptive Δ-MOCK, is an adaptive parameter control method. Our approach is different in the sense that the parameter to be tuned governs properties of the optimization problem itself (by expanding the search space), rather than properties within the algorithm such as the population size or mutation probability. Arguably, two of the most similar methods to ours are the ARGOT strategy [16] and the delta coding [18]. The ARGOT strategy uses operators that allow both reduction and expansion of the search space, through adaptation of the number of bits used to represent the gene. On the other hand, delta coding is a scheme that utilizes multiple restarts of the algorithm, adjusting the search space based on the performance of the previous run.

Similar to other adaptive parameter control methods, our approach relies on feedback from the search to trigger a change in a parameter value. Typical metrics to quantify feedback include the quality of the best solution in the population or the quality of the population as a whole. In the presence of multiple objectives, quantifying feedback (or algorithm performance) is not straightforward as the goal is to evolve a set of diverse trade-off solutions [21]. In our case, feedback is based on the hypervolume metric [21] - well accepted in the community due to its Pareto compliant property.

While the hypervolume has become a popular metric in both parental and environmental selection operations in a multiobjective optimization algorithm (e.g. IBEA [20] and SMS-EMOA [2]), there is limited work on using the metric to measure convergence to the Pareto front to drive evolution. This approach was first considered in [17], and later incorporated into NSGA-II in [12], where it is used as a mechanism to increase the population size if there is no significant change in the hypervolume (though no definition of this change is provided). Other work uses the fitness values directly to detect a change in performance, either by statistical hypothesis testing [15] or by monitoring the average fitness of the best found solutions [3]. In the latter work, a drop in this average fitness triggers the use of hypermutation (a strategy also employed in this paper) to try and react to the environmental change responsible for this.

3 Δ-MOCK AND ITS HYPERPARAMETER δ

The multi-objective clustering algorithm Δ-MOCK [6] minimises two objectives: intra-cluster variance and connectivity. The former objective emphasizes cluster compactness and is minimised when every data point is in its own separate cluster; the latter considers preservation of nearest neighbour relationship, and is minimised when all data points are together in a single cluster.

To accommodate clusters associated with these complementary objectives, clustering solutions in the original MOCK algorithm were represented using the locus-based adjacency encoding [13], which is illustrated in Figure 1a. In Δ-MOCK, this encoding mechanism was further refined using knowledge derived from the Minimum Spanning Tree (MST) of a dataset. During the precomputation phase of the algorithm, the MST of the data is determined, and each link (or edge) between two nodes (data points) is assigned a *degree of interestingness* (DI) value, as indicated in Figure 1a. The DI of the link $i \rightarrow j$ is defined as

$$DI(i \rightarrow j) = \min\left\{nn_i(j), nn_j(i)\right\} + \frac{\sigma(i,j)}{\sigma_{max}}, \tag{1}$$

where $nn_i(j)$ is the ranking of data point j in data point i's nearest neighbours. The (Euclidean) distance between i and j is given by $\sigma(i,j)$, and is divided by the maximum distance in the dataset (σ_{max}) to ensure this term is in the range $[0,1]$.

A ranking is created for all links in the MST by sorting on their DI value. The hyperparameter δ then specifies the percentage of the least interesting links to fix for all individuals, effectively reducing the genotype and thus the search space to the non-fixed set of links. This approach is illustrated in more detail in Figure 1a, where we use *'?'* to indicate those links that are not fixed (and therefore define the available search space). Considering Figure 1c, it is clear that $\delta = 0$ corresponds to MOCK's original encoding (maximum resolution), whereas $\delta = 100$ collapses the search space to a single point, with all data points assigned to the same cluster (minimum resolution). At $\delta = 50$ (see Figure 1b), half of the least interesting links in the MST have been fixed, creating a partial solution with $P = 6$ components. The reduced genotype then serves to represent

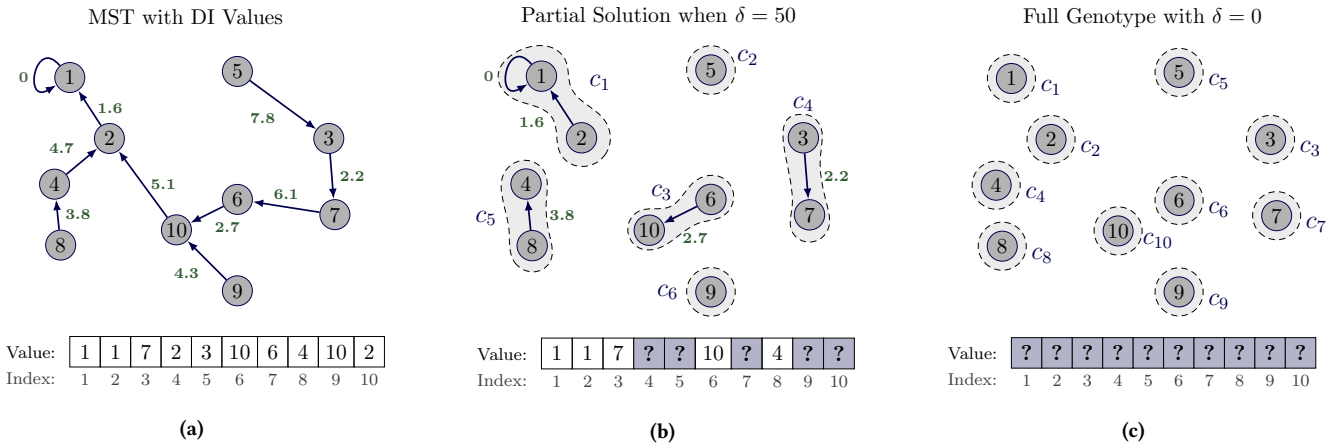


Figure 1: An example dataset with $N = 10$ data points is presented. In (a), the MST is shown, and is equivalent to $\delta = 100$. The DI value of each link is displayed next to the corresponding edge. In (b), the $P = 6$ components as defined by $\delta = 50$ can be seen, where half of the least interesting links have been fixed. This produces a reduced genotype of length $\Gamma = 5$. In (c), there are no fixed links when $\delta = 0$, resulting in a full-length genotype $P = N$.

combinations of these components. As the value of δ fixes a percentage of the genotype and thus links in the MST, it is clear that the optimal value of δ will differ for different datasets.

The initialization routine of Δ-MOCK is specialised to create a close initial approximation of the Pareto front. This is achieved by creating individuals for the initial population by removing the k most interesting links, where k is randomly sampled from $\{2, 3, \ldots, k_{max}\}$ and k_{max} is twice the expected number of clusters. As removal of these links is only permitted for the unfixed links, δ determines what individuals can be generated by this routine.

The neighbourhood-biased mutation operator used by Δ-MOCK calculates the mutation probability individually for every gene, where for a link $i \rightarrow j$ encoded in a gene the probability is:

$$p_m(i \rightarrow j) = \frac{1}{\Gamma} + \left(\frac{nn_i(j)}{\Gamma}\right)^2, \text{ with } \Gamma = \left(1 - \frac{\delta}{100}\right) N. \qquad (2)$$

Here N gives the number of points in the dataset, and Γ indicates the length of the (unfixed) genotype associated with a given value of δ. When selected for mutation, the link $i \rightarrow j$ is replaced with the link $i \rightarrow h$ where h is randomly chosen from the data point i's L nearest neighbours, where L is typically quite small. Consistent with the original MOCK algorithm, Δ-MOCK uses uniform crossover as its crossover operator.

4 ADAPTIVE Δ-MOCK

Algorithm 1 outlines the proposed adaptive version of Δ-MOCK; in particular, the additions introduced to Δ-MOCK can be found in Lines 10-12. A brief overview of the most relevant parts of the original Δ-MOCK have been given in the previous section; for further details the reader is referred to [6].

Here, we limit ourselves to strategies capable of a incremental increase in resolution, as defined by a step-wise decrease in the value of the hyperparameter δ. Specifically, we consider the situation of a given set of T levels of resolution, with the minimum resolution level defined by δ_{High} and the maximum resolution described by δ_{Low}. We consider search strategies that start at δ_{High} and incrementally decrease the δ value from δ_{High} to δ_{Low}. The timing of the value change is determined by the trigger mechanism used, with the search strategy employed determining how Δ-MOCK reacts to this change. Detailed descriptions of different trigger mechanisms and search strategies are provided in the following.

Algorithm 1 Adaptive Delta-MOCK

Require: Dataset, δ, G_{max}
1: *format_data*() // Standardize, etc.
2: *precomputation*() // Distance matrix, MST, DI, etc.
3: $\mathcal{P}$ = *create_initial_pop*($|\mathcal{P}|$)
4: HV_0 = *comp_hypervolume*($\mathcal{P}$)
5: **for** gen = 1 **to** G_{max} **do**
6: $\hat{\mathcal{P}}$ = *parental_selection*($\mathcal{P}$)
7: $\mathcal{P}'$ = *genetic_operators*($\hat{\mathcal{P}}$)
8: $\mathcal{P}$ = *environmental_selection*($\mathcal{P} \cup \mathcal{P}'$)
9: HV_{gen} = *comp_hypervolume*($\mathcal{P}$)
10: **if** *trigger_mechanism*() is **True then**
11: Reduce δ and recompute auxiliary variables
12: Activate search strategy
13: $\mathcal{P}$ = *Pareto_nondominated*($\mathcal{P}$)

4.1 Trigger Mechanisms

As the optimal δ value is unknown, criteria are needed to decide when it is advantageous (if at all) to reduce δ. To determine at which generations this occurs, one adaptive and two baseline trigger mechanisms are considered:

(1) *Random* — Randomly samples the generation at which to trigger a decrease in δ
(2) *Interval* — Tries to ensure that Δ-MOCK has an approximately equal number of generations at each δ value explored

(3) *HV* — Hypervolume-based mechanism, which uses the hypervolume of the current population to determine whether the search is stagnating and therefore if a decrease in δ is required

If the population has converged to the Pareto front defined by the current δ, expanding the search space by reducing δ enables Δ-MOCK to use its remaining evaluations more effectively. The hypervolume can be used to evaluate this convergence, as it is a measure of the volume of the objective space dominated by a population (or set of objective vectors) [21]. By tracking the hypervolume for each generation (using the worst possible objective values as the reference point), it can be seen when the population begins to converge. As a reference is required to identify convergence, we obtain this by calculating ΔHV in the first $G_{max}/10$ generations, where G_{max} is the number of generations. A moving average gradient of the hypervolume is maintained during evolution. When this average falls significantly below the reference gradient (< 0.1 of the reference gradient), this indicates that the rate of improvement has become negligible and thus a decrease of δ can be triggered.

To determine effectiveness of the hypervolume trigger mechanism, we compare it to two baseline approaches: random and interval. Both approaches are prescriptive in that the total number of desired triggers $T-1$ (as no trigger is needed for the initial resolution) is specified, and they proceed by generating numbers specifying the generation(s) at which to decrease δ. The random approach simply chooses a random generation, whereas the interval approach attempts to ensure that the algorithm has an approximately equal amount of time at each genotype length. More specifically, the random trigger mechanism randomly samples $T-1$ generations from $\{1, \ldots, G_{max}\}$ to trigger the incremental reduction in δ. The interval mechanism selects these $T-1$ generations by sampling one number from $T-1$ equal intervals in the range $\{G_{max}/10, \ldots, G_{max} - (G_{max}/10)\}$ to ensure each δ value has an approximately equal number of generations. In contrast to this, the hypervolume mechanism is adaptive to current performance and determines when a decrease in δ is needed. For comparability, we prevent it from increasing the resolution more than $T-1$ times.

In Algorithm 1, Line 10, the condition depends on the trigger mechanism used. For the two baselines mechanisms, this is simply a test of whether the current generation is in the list of values generated by the mechanism, whereas for the hypervolume-based trigger mechanism it depends on the calculation described above.

4.2 Search Strategies

The new search space defined by the reduced δ value should be explored rapidly to discover partitions now available with the new components, particularly as a change in δ may occur late in the search. Ideally, focus should be on this new region to avoid repeated or unnecessary evaluations; for this, five strategies are considered:

(1) TH_{all} — Triggered hypermutation, applied to all genes in the (reduced) genotype for a single generation
(2) TH_{new} — Triggered hypermutation, applied to just the genes of newly unfixed links for a single generation
(3) *RO* — Reinitialized offspring, where Δ-MOCK's specialized initialization is used to generate the offspring

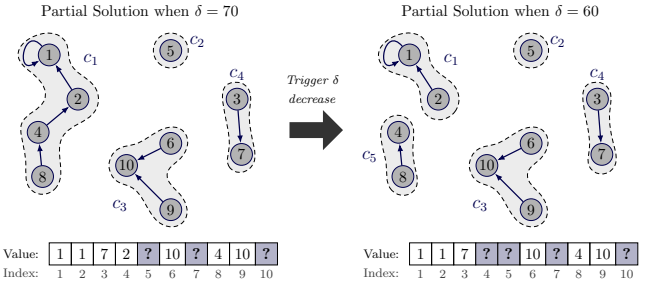


Figure 2: Illustration of the example dataset being run with $\delta = 70$. Triggering a decrease in δ reduces it to 60, unfixing the next most interesting link. This splits one of the components, creating new possible partitions of the data.

(4) *FM* — Fair mutation, where a number of individuals are created to explore each of the newly unfixed links
(5) *CO* — Carry on, a baseline strategy where no other changes are made beyond the decrease in δ

In both the TH_{all} and TH_{new} strategies, a hypermutation rate is used for a single generation only. Similarly, the *RO* strategy affects the generation of a single set of offspring only. The fair mutation (*FM*) approach extends the strategy introduced in [1], which was used for a binary genotype to explore the space of solutions including newly introduced genes. The offspring are created by generating an equal number of individuals for each new gene in the genotype. As all individuals will have the same value in the newly unfixed links (the edge found in the MST), individuals will be generated separately for each particular gene by setting the value to be self-connecting, such that the link is absent. As a result, new components are presented that can be merged, exploring previously unseen partitions. The gene set to be self-connecting for each generated individual is 'protected' for a defined number of generations, during which this value cannot be changed. The mutation probability for all other links in the genotype is raised in order to rapidly explore the effect that this new gene has.

Figure 2 shows the partial solution obtained when $\delta = 70$, fixing all but the 3 most interesting links using the DI values from Figure 1a. While this simplifies the optimization problem, there may be fixed links that connect separate clusters which, by reducing δ, become unfixed and are now available in the search. However, this longer genotype increases the computational cost of the fitness evaluation.

In the illustration, δ is reduced to 60 which unfixes the next most interesting link. With the TH_{all} scheme, the hypermutation rate would be multiplied to each of the probabilities calculated (using Equation 2) for all 4 genes of the reduced genotype, whereas TH_{new} just multiplies the mutation probability (p_m) of the new gene at index 4. In this example with just a single new gene, the *FM* scheme would generate all offspring to have a self-connecting link at index 4, using a raised mutation rate for the rest of the genotype.

4.3 Impact on Precomputation

An advantage of the δ hyperparameter, beyond simplifying the optimization, is to reduce the computation required by the evaluation function via precomputation. Precomputed values for both objectives can be calculated for the cluster components obtained when $\delta > 0$, such as the 6 components present in Figure 1b. This simplifies the evaluation function to a simpler calculation, defined by combining components encoded by the reduced genotype; details for this method can be found in [6]. A change in δ requires this precomputation to be performed again, as new components are available in the search (as shown in Algorithm 1, Line 11).

5 EXPERIMENTAL STUDY

This section describes the experimental setup designed to compare the efficacy of the search strategies within Adaptive Δ-MOCK using the different trigger mechanisms. The parameters used for each method tested are provided, followed by an analysis of the results with an explanation of observed behaviour.

5.1 Experimental Design

Δ-MOCK and its strategies were implemented in Python, in accordance with the previously published code[1], where the datasets used here can also be found. All search strategies and Δ-MOCK configurations have been run with 30 independent runs for each trigger mechanism. The design of the experiment and parameters used are outlined below and in Table 1.

Table 1: Experimental Setup

Configuration	Adaptive δ?	Trigger	Starting δ	Final δ
Δ-MOCK ($sr1$)	No	–	$sr1$	$sr1$
Δ-MOCK ($sr5$)	No	–	$sr5$	$sr5$
Δ-MOCK with $\{CO, FM, TH_{all}, TH_{new}, RO\}$	Yes	*Random*	$sr1$	$sr5$
		Interval	$sr1$	$sr5$
		HV	$sr1$	?*

∗: Final genotype length depends on performance

Standard parameters for Δ-MOCK are used in all seven configurations: The number of generations, $G_{max} = 100$; the population size $|\mathcal{P}| = 100$; the neighbourhood parameter $L = 10$; and, the crossover probability $p_r = 1.0$. In both the TH_{all} and TH_{new} strategies, a hypermutation rate of $500 \times p_m$ is used, while the *FM* strategy uses a raised mutation rate of $50 \times p_m$ for the non-protected genes, and the gene protected in each of the individuals becomes unprotected after 3 generations.

To enable setting an appropriate δ for datasets of different scales, work in [6] introduced notation where the genotype length is a function of the square root of the number of data points ($\sqrt{N}$), allowing δ to be calculated. For a desired genotype length of $\alpha\sqrt{N}$, the notation is $sr\alpha$. For this experiment, $\delta_{\text{High}} = sr5$ and $\delta_{\text{Low}} = sr1$ are used; these were found to be generally good and poor values respectively of δ for the datasets used in [6]. To effectively compare strategies to the unmodified Δ-MOCK and assess their ability to produce suitable partitions of the data, constraints ensure the maximum (reduced) genotype length available is equal for all methods. The experiment is created as follows: Δ-MOCK, with no modifications, is run separately at both δ_{High} and δ_{Low}. All other strategies are restricted to use $T = 5$ resolution levels from the set $\{sr1, sr2, sr3, sr4, sr5\}$, beginning at $\delta_{\text{Low}} = sr1$ and expanding the genotype incrementally according to the trigger mechanism employed.

A total of 43 datasets have been used in our experiments, 35 synthetic and eight real-world datasets. The synthetic datasets contain clusters of arbitrary shape [6] in a range of configurations. Each dataset has a dimensionality $D \in \{20, 50, 100, 150, 200\}$ and, for each of these, a number of clusters $c \in \{10, 20, 40, 60, 80, 100, 120\}$ for a total of 35 datasets with an average of 5,392 data points. A dataset with 100 dimensions and 40 clusters will be referred to using the format $100D{-}40c$. The remaining eight are large, real-world datasets curated by [6], which consists of the latitude and longitude of crime locations ($D = 2$), with a lower number of clusters ($c \in \{10, 11, 12\}$) and an average of 30,685 data points. These datasets will be referred to as *UKC1*, ..., *UKC8*.

The Adjusted Rand Index (ARI) is used to measure the quality of the partitions produced, as labels are available for these datasets. The ARI is in the range [∼0, 1], where 1 is a perfect clustering and 0 is random labelling [14]. The execution times recorded for the search strategies include recomputation of the precomputed variables, and the hypervolume calculation for that trigger mechanism, ensuring a fair assessment of the impact of these modifications.

5.2 Results

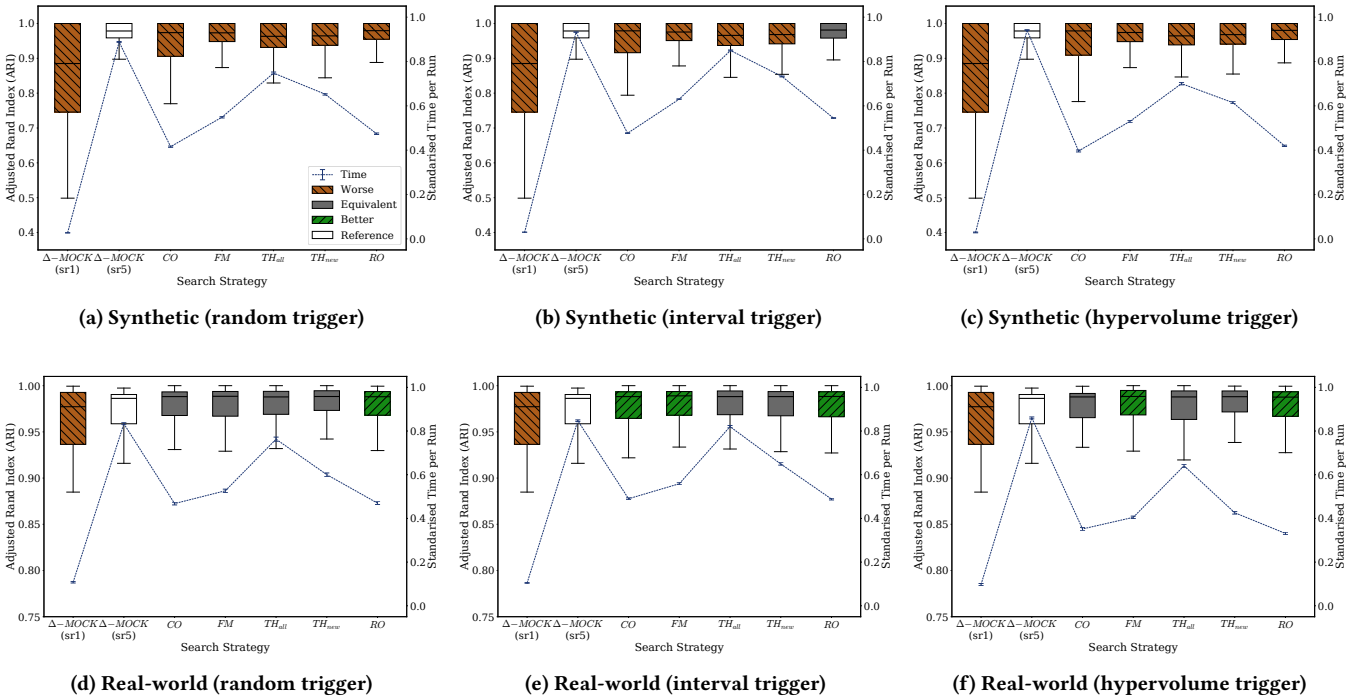
Multi-objective clustering methods return a set of solutions corresponding to a range of different trade-offs and numbers of clusters. The ARI values used in our comparison are sets of 30 values corresponding to the best solution found in each run – this is to assess the potential of each method to generate good solutions. As some Pareto optimal solutions will have too many or too few clusters (leading to poor ARI values), mean or median values across individual approximation sets would be of less interest here.

The performance (ARI) of each strategy and trigger mechanism, and the time taken to run (standardized between 0 and 1 for each dataset) is compared to Δ-MOCK using a fixed genotype length of $\Gamma = sr5$. Statistical testing was performed using the Wilcoxon signed-rank test with a $\alpha = 0.05$ significance level. In Figure 3 any strategy found to be statistically significantly worse is filled in orange, equivalent in grey, and significantly better than Δ-MOCK ($sr5$) are filled in green.

Figures 3a-3c aggregate results for the synthetic datasets. Of the strategies tested, the best and most similar performance to the baseline Δ-MOCK is obtained using the *RO* strategy, and is achieved at a 46.58%, 41.56%, and 55.42% average reduction in computation time for the random, interval, and hypervolume trigger mechanisms respectively. As the hypervolume mechanism does not require expanding to a $\Gamma = sr5$ genotype length, largely equivalent performance can be obtained at $\Gamma < sr5$ at a more significant reduction in computation.

Using Δ-MOCK with a fixed $\Gamma = sr1$ is clearly too restrictive for most problems as performance is poor, though this is associated

[1] https://github.com/garzafabre/Delta-MOCK



(a) Synthetic (random trigger) **(b) Synthetic (interval trigger)** **(c) Synthetic (hypervolume trigger)**

(d) Real-world (random trigger) **(e) Real-world (interval trigger)** **(f) Real-world (hypervolume trigger)**

Figure 3: ARI (left y-axis) of each strategy for the synthetic datasets, with a line showing the average standardized run time (right y-axis) for each strategy. All configurations are compared to Δ-MOCK *(sr5)* using the Wilcoxon signed-rank test. Statistically significantly worse strategies are filled in orange, equivalent in grey, and better than Δ-MOCK *(sr5)* are filled in green. Δ-MOCK *(sr1)* is found to be consistently worse, and all other strategies are found to be equivalent or better regardless of the trigger mechanism used, with the *RO* strategy found to be better across all 3 trigger mechanisms.

with the fastest computation time. Relative to the poor performance of *CO*, the efficacy of all other search strategies can be seen. Without any modifications beyond a change in δ, Δ-MOCK is unable to rapidly search this new space and significantly change the values of the newly introduced genes (the uniform crossover operator is ineffective as the values in these genes are the same for all individuals).

To understand the results in more detail, an empirical attainment function (EAF) difference plot compares performance of two methods in the objective space. The shading in these plots indicates the difference in the probability of a method obtaining a point in that region of the search space in favour of the method to which it is compared [11]. Additionally, the objective values of the individual with the best ARI found by either method are shown in both figures.

In Figure 4b, the behaviour of the *CO* and TH_{all} strategies is compared (results shown are from runs using the interval trigger mechanism), ensuring any differences observed solely capture the effect of an increased mutation rate. This comparison indicates a clear tendency of the mutation operator to generate offspring that favour the intra-cluster variance. This behaviour can be understood in terms of MOCK's mutation scheme. As the mutation operator chooses to replace a link randomly from a data point's nearest neighbours, this is more likely to create an intra-cluster than an inter-cluster link, increasing the number of clusters and thus favouring the intra-cluster variance objective. This intrinsic bias is likely to explain the (relatively) poor performance of the two *TH* strategies (and to a lesser extent, *FM*), compared to the *RO* strategy, which puts no additional emphasis on mutation.

Figure 4a reveals that Δ-MOCK ($sr5$) is better at optimizing the intra-cluster variance objective compared to the *RO* strategy, which obtains slightly better results for the connectivity objective. This indicates that the introduction of genetic material by the initialization routine helps to counter a bias of the mutation operator, as explained above.

Figures 3d-3f show an aggregation of the results for the real datasets. Although Δ-MOCK ($sr1$) was again the worst configuration tested, good clustering solutions were still found as the larger size and low complexity of the data meant that the resolution obtained using $\Gamma = sr1$ is almost adequate. All tested strategies display equivalent or better performance over Δ-MOCK ($sr5$). Again, it is clear from Figure 5 that increasing the mutation rate removes focus from the interesting region of the objective space, as the average number of clusters defined by the individuals in the final population far exceeds the true number. While the increased mutation rate has allowed the search space to be explored more rapidly, occasionally identifying better partitions, this has come at the cost of the population as a whole becoming worse on average.

The *RO* strategy was again the fastest tested, with a 43.63%, 42.34%, and 61.49% average reduction in computation time for the

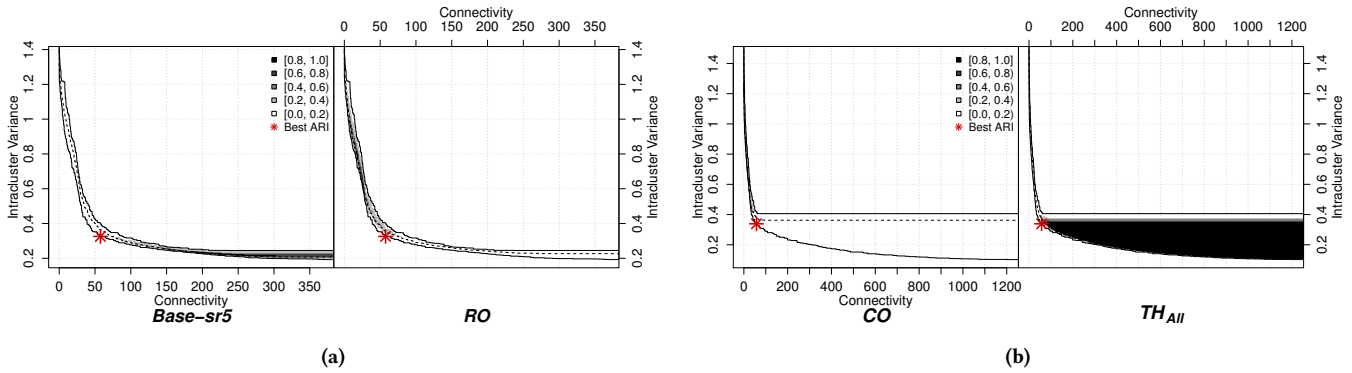


Figure 4: Visualisations of the difference between the empirical attainment functions of two methods. The asterisk (∗) shows the objectives values of the individual with the best ARI found from either method. Darker shading indicates that that method has a higher probability of finding solutions in that region of the search space compared to the other method. In (a), we compare Δ-MOCK *(sr1)* with the *RO* strategy, showing a bias of the former to the intra-cluster variance objective, and of the latter to connectivity. In (b), the only difference between the two methods shown is the increased mutation rate, which clearly favours the intra-cluster variance.

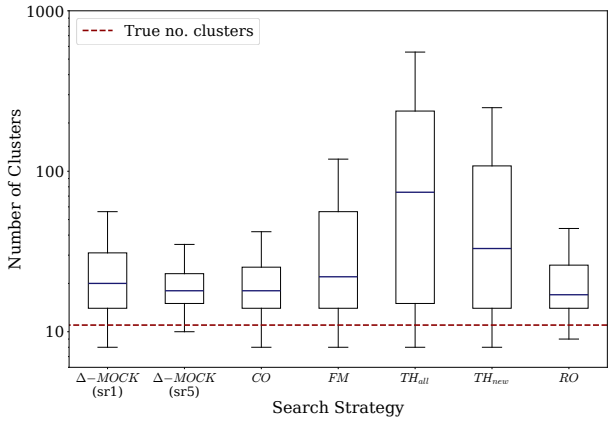


Figure 5: The number of clusters generated by each method tested for the *UKC5* dataset, with the true number of clusters shown by the dashed horizontal line, highlighting the large differences in the populations produced.

random, interval, and hypervolume trigger mechanisms respectively, making the hypervolume-based trigger mechanism even more computationally efficient with no drop in performance. For these datasets the performance over Δ-MOCK is actually increased, with significantly better performance found across all three trigger mechanisms. This indicates that the methods are able to benefit from the focused search in the most relevant search space.

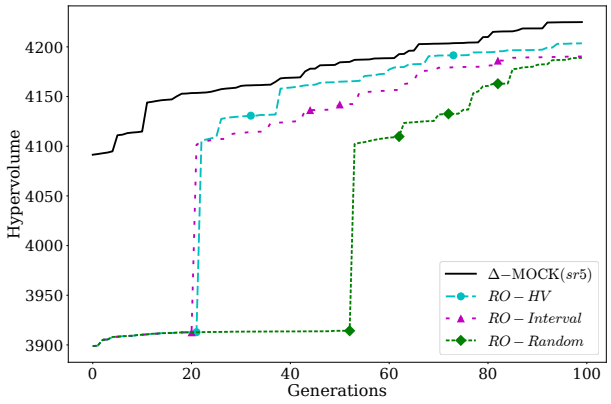
Looking at the hypervolume of the population during evolution provides insight into how the trigger mechanisms behave individually. Figure 6 shows generational plots for a single run on two datasets. In Figure 6a, an example with the $50D - 60c$ dataset is shown where $\Gamma = sr1$ is too restrictive, yet after the first decrease in δ the search space is opened sufficiently to produce a population that appears nearly equivalent (in terms of the achievable hypervolume) to the initial population of unmodified Δ-MOCK ($sr5$). Subsequent triggers have a less pronounced effect, indicating that the links introduced beyond the first trigger are not useful. This is identified by the hypervolume trigger mechanism which does not expand to $\Gamma = sr5$ (only three trigger points are recorded). The method clearly achieves a higher hypervolume than the random trigger mechanism which is hampered by spending more than half of the available generations at a poor δ value.

Figure 6b shows results for the $200D{-}20c$ dataset. Here, the early triggers have a less pronounced effect, suggesting that there are no significant interesting links beyond the set $\Gamma = sr1$. For the hypervolume-based trigger mechanism, only two triggers are recorded. The smaller genotype enables greater focus on more interesting combinations of cluster components, allowing the mechanism to obtain the best hypervolume with a final genotype length of $sr3$.

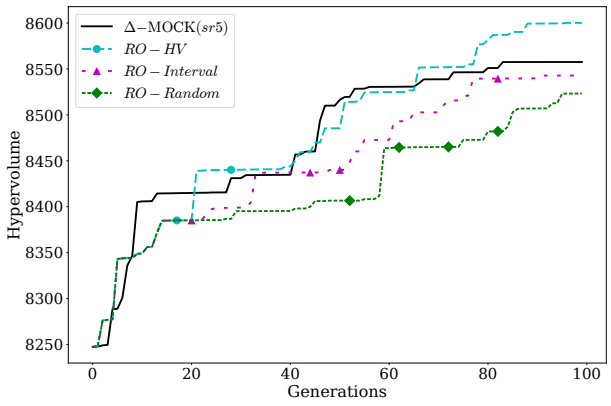
6 CONCLUSION AND FUTURE WORK

Our results indicate that the tuning of a hyperparameter controlling the length of the encoding for evolutionary clustering has potential in focusing the search and reducing computational costs. The current setup has some limitations, however. Specification of a starting point is required, and the resolution can only be adjusted in a single direction (expansion of the search space). Further study is warranted to improve flexibility of the strategies. With additional development, there is scope for some of these strategies to generalize to other cluster encodings.

The ability of δ to adapt during run-time is intuitively most useful for large, complex clustering problems. This was confirmed by the results of the *RO* strategy: even when starting with a generally restrictive value of δ, competitive performance was obtained for a significant reduction in computation time. The absence of significant computational overhead associated with the *RO* strategy



(a) Generational plot for the 50D−60c dataset



(b) Generational plot for the 200D−20c dataset

Figure 6: Generational plots for two synthetic datasets comparing Δ-MOCK *(sr5)* with the *RO* strategy run with each trigger mechanism. The graphs show a single run to illustrate the behaviour of the trigger mechanisms. The points highlight the generations where δ was decreased.

and its consistent performance indicate that its use in Adaptive Δ-MOCK allows the application of Δ-MOCK to ever larger problems at a reduced computational cost.

The current bias of MOCK's mutation operator is an issue for generating a population of candidates focused on the area of the objective space that is most interesting for good partitions of the data, and has hampered the performance of several strategies explored here. Future work may consider an appropriate modification to this, or its complete removal. The success of the *RO* strategy confirms that Δ-MOCK's specialized initialization clearly introduces most of the genetic material useful for clustering, and mutation may therefore be of less importance. On the other hand, it is possible that refinement of the crossover operator to be sensitive to search space changes may further encourage rapid convergence.

ACKNOWLEDGMENTS

We thank Manuel López-Ibáñez for assistance with the EAF plots; Cameron Shand acknowledges PhD funding support from the EPSRC Manchester Centre for Doctoral Training in Computer Science (EP/I028099/1).

REFERENCES

[1] Richard Allmendinger and Joshua Knowles. 2010. Evolutionary optimization on problems subject to changes of variables. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6239 LNCS, PART 2 (2010), 151–160.

[2] Nicola Beume, Boris Naujoks, and Michael Emmerich. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181, 3 (2007), 1653–1669.

[3] Helen G Cobb. 1990. An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments. *Technical Report* (1990). https://doi.org/doi=10.1.1.79.4834

[4] Michel Delattre and Pierre Hansen. 1980. Bicriterion cluster analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 4 (1980), 277–291.

[5] Ágoston E Eiben, Robert Hinterding, and Zbigniew Michalewicz. 1999. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 3, 2 (jul 1999), 124–141. https://doi.org/10.1109/4235.771166

[6] Mario Garza-Fabre, Julia Handl, and Joshua Knowles. 2017. An Improved and More Scalable Evolutionary Approach to Multiobjective Clustering. *IEEE Transactions on Evolutionary Computation* V (2017), 1–1. https://doi.org/10.1109/TEVC.2017.2726341

[7] Julia Handl and Joshua Knowles. 2004. Evolutionary multiobjective clustering. In *International Conference on Parallel Problem Solving from Nature*. Springer, 1081–1091.

[8] Julia Handl and Joshua Knowles. 2007. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation* 11, 1 (2007), 56–76. https://doi.org/10.1109/TEVC.2006.877146

[9] Adán José-García and Wilfrido Gómez-Flores. 2017. Evolutionary Clustering Using Multi-prototype Representation and Connectivity Criterion. In *Mexican Conference on Pattern Recognition*. Springer, 63–73.

[10] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston E Eiben. 2015. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation* 19, 2 (2015), 167–187.

[11] Manuel López-Ibáñez, Luis Paquete, and Thomas Stützle. 2010. Exploratory Analysis of Stochastic Local Search Algorithms in Biobjective Optimization. In *Experimental Methods for the Analysis of Optimization Algorithms*, Thomas Bartz-Beielstein, Marco Chiarandini, LuÃıs Paquete, and Mike Preuss (Eds.). Springer, Berlin, Germany, 209–222. https://doi.org/10.1007/978-3-642-02538-9_9

[12] Saúl Zapotecas Martínez, Edgar G Yánez Oropeza, and Carlos A Coello Coello. 2011. Self-adaptation techniques applied to multi-objective evolutionary algorithms. In *International Conference on Learning and Intelligent Optimization*. Springer, 567–581.

[13] YoungJa Park and ManSuk Song. 1998. A Genetic Algorithm for Clustering Problems. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*. Morgan Kaufmann, University of Wisconsin, Madison, Wisconsin, USA, 568–575.

[14] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.

[15] Hendrik Richter. 2009. Detecting change in dynamic fitness landscapes. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 1613–1620.

[16] Craig G. Shaefer. 1987. The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 50–58. http://dl.acm.org/citation.cfm?id=42512.42520

[17] Heike Trautmann, Uwe Ligges, Jörn Mehnen, and Mike Preuss. 2008. A convergence criterion for multiobjective evolutionary algorithms based on systematic statistical testing. In *International Conference on Parallel Problem Solving from Nature*. Springer, 825–836.

[18] Darrell Whitley, Keith Mathias, and Patrick Fitzhorn. 1991. Delta coding: An iterative search strategy for genetic algorithms. In *ICGA*, Vol. 91. 77e84.

[19] David H. Wolpert and William G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (Apr 1997), 67–82. https://doi.org/10.1109/4235.585893

[20] Eckart Zitzler and Simon Künzli. 2004. Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature*. Springer, 832–842.

[21] Eckart Zitzler and Lothar Thiele. 1998. Multiobjective optimization using evolutionary algorithms — A comparative case study. In *Parallel Problem Solving from*

Nature — PPSN V, Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 292–301.