2013 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 22–25, 2013, SOUTHAMPTON, UK

A NOVEL HEURISTIC MEMETIC CLUSTERING ALGORITHM

B.G.W. Craenen[2,1*] *A.K. Nandi*[1,2†] *T. Ristaniemi*[2]

[1]Brunel University
Uxbridge, Middlesex, UK
{Bart.Craenen|Asoke.Nandi}@brunel.ac.uk

[2]University of Jyväskylä
Jyväskylä, Finland
Tapani.Ristaniemi@jyu.fi

ABSTRACT

In this paper we introduce a novel clustering algorithm based on the Memetic Algorithm meta-heuristic wherein clusters are iteratively evolved using a novel single operator employing a combination of heuristics. Several heuristics are described and employed for the three types of selections used in the operator. The algorithm was exhaustively tested on three benchmark problems and compared to a classical clustering algorithm (k-Medoids) using the same performance metrics. The results show that our clustering algorithm consistently provides better clustering solutions with less computational effort.

***Index Terms*—** Clustering, Memetic Algorithms, Heuristics

1. INTRODUCTION

Clustering is a data and signal processing task where the objective is to determine a finite set of categories, called clusters, to describe a dataset according to the similarities among its objects [1]. The applications of clustering are manifold and range from market segmentation [2] and image processing [3] to document categorisation and web mining [4]. More recently clustering has gained prominence for signal processing in the field of bioinformatics [5].

Clustering as an optimisation problem maximises the homogeneity within clusters while maximising the heterogeneity between clusters [3]. It strives to ensure that (data) objects belonging to the same cluster are more similar to each other and more different to those belonging to other clusters. Measuring similarity and dissimilarity is usually tackled indirectly. Distance measures are used for quantifying the degree of dissimilarity among data objects in such a way that more similar data objects have lower dissimilarity values [6]. Several distance measures can be deployed for clustering [3], but here we use the Euclidean distance measure.

*This research was supported by TEKES (Finland) grant 40334/10 Machine Learning for Future Music and Learning Technologies (MUSCLES).

†Asoke K. Nandi would like to thank TEKES for the award of the Finland Distinguished Professorship

Clustering techniques can be broadly divided into three main types [3]: overlapping (non-exclusive), partitional, and hierarchical. The former two types become related in hierarchical clustering, where a nested sequence of partial clusterings is both hierarchical and partitional. In this study we consider only partitional clustering.

Clustering is deemed one of the most difficult and challenging problems in Machine Learning. This is due mostly to its unsupervised nature. From a theoretical perspective, clustering is formally considered as a particular kind of NP-hard grouping problem [7]. This has stimulated the search for efficient approximation algorithms, including not only the use of *ad hoc* heuristics for particular classes or instances of problems, but also the use of general-purpose meta-heuristics [8]. Particularly Evolutionary Algorithms (EAs), and as a subset Memetic Algorithms (MAs), are widely held to be effective on these kinds of NP-hard problems. They are able to provide near-optimal solutions to such problems in reasonable time. Many clustering EAs and MAs have been proposed in literature [9].

MAs represent one of the recent growing areas of research in Evolutionary Computation and Machine Learning. They represent a synergy of evolutionary, individual, and local search learning. In literature, MAs are sometimes referred to as Baldwinian EAs, Lamarckian EAs, Cultural Algorithms, or Genetic Local-Search Algorithms. MAs, however, sometimes suffer from the *algorithm-of-many-parts* problem, also called *Memetic Overkill* [10]. When MAs are afflicted by Memetic Overkill it becomes difficult to identify which parts of the algorithm contribute toward finding solutions, even whether some parts hamper finding solutions. Memetic Overkill is best avoided by rigorously limiting the constituting parts of the MA, adding new ones only when their effects on the whole algorithm is shown to be beneficial.

This study presents a novel clustering algorithm based on the MA meta-heuristics. We set out to develop an algorithm that is both easy to understand and use, using techniques that provide robust performance with as few parameters required as possible. The idea behind this approach is to provide a benchmark general-purpose algorithm that is easy to tailor to specific problems and further research. The novel aspect

978-1-4799-1180-6/13/$31.00

of the algorithm is the single search operator. It uses three types of local-search heuristics, with several options available for each. Different local-search heuristics constitute different variants of the algorithm, which can be used to exploit features of different problems and problem instances. In this study we demonstrate experimentally that our algorithm consistently outperforms the commonly used k-Medoids algorithm on three benchmark datasets on shared effectiveness and efficiency metrics. As far as the authors are aware, no such algorithm or systematic investigation of these heuristics within a single MA framework has been published before.

The remainder of this study is then organised as follows. Section 2 provides a definition of the problem, and a notation used throughout the paper. Section 3 describes the algorithm and the heuristics used. In section 4 we shortly describe the three benchmark datasets used. Section 5 presents our experimental method ans setup, with the results presented in section 6. Finally, a conclusion is provided in section 7.

2. PROBLEM DEFINITION

In this section we will provide a formal definition of the clustering problem and introduce a notation used in the rest of the paper. Where super- or subscripts are used, we will reuse i, j, l, m. No meaning is implied where these are carried over between definitions. As is customary in literature, bold-faced variables indicate sets, with $|\mathbf{Y}|$ defined as the number of elements in set $\mathbf{Y}$, i.e., its size.

A n-dimensional feature or attribute vector $x = (x^1, x^2, \ldots, x^n)$ is called a data object, with x^i, $(i = 1, 2, \ldots, n)$ the i-th feature, attribute, or data value. A dataset $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ has $N = |\mathbf{X}|$ data objects, with x_i, $(i = 1, 2, \ldots, N)$ the i-th data object. A clustering $\mathbf{C}$ is a collection of data object subsets $\mathbf{c}_i$ $(i = 1, 2, \ldots, k)$, called clusters, with $k = |\mathbf{C}|$ traditionally reserved to denote the size of the clustering; the number of clusters, $|\mathbf{c}|$ denotes the size of cluster $\mathbf{c}$, i.e., the number of data objects in the cluster. A cluster is not allowed to be empty: $\mathbf{c} \neq \emptyset$ or $|\mathbf{c}| \neq 0$, and the conjunction of all clusters contains all data objects of the dataset: $\mathbf{c}_1 \cup \mathbf{c}_2 \cup \ldots \cup \mathbf{c}_k = \mathbf{X}$. In a non-overlapping clustering, all clusters are mutually disjunct: $\mathbf{c}_i \cap \mathbf{c}_j = \emptyset$, for $i \neq j$. If the mutual disjunction criteria is relaxed, clustering $\mathbf{C}$ is said to be overlapping. Here we only consider non-overlapping clustering.

The Euclidean distance metric $d(x_i, x_j)$ is used to calculate the distance between two data objects x_i and x_j. $d(x, \mathbf{c})$ is used to indicate the set of distances between data object x and all data objects in cluster $\mathbf{c}$: $d(x, \mathbf{c}) = \{d(x, x'_1), d(x, x'_2), \ldots, d(x, x'_{|\mathbf{c}|})\}$ with $x' \in \mathbf{c}$. The inner-cluster distance of cluster $\mathbf{c}$ is used as the basis for the fitness function and most of the heuristics, and is calculated as follows:

$$D(\mathbf{c}) = \sum_{l=1}^{|\mathbf{c}|-1} \sum_{m=l+1}^{|\mathbf{c}|} d(x_l, x_m) \quad (1)$$

where $x_l \in \mathbf{c}$ and $x_m \in \mathbf{c}$. If the inner-cluster distance is calculated for a clustering $\mathbf{C}$, it produces a set containing the inner-cluster distances of all clusters in the clustering: $D(\mathbf{C}) = \{D(\mathbf{c}_1), D(\mathbf{c}_2), \ldots, D(\mathbf{c}_k)\}$. The inner-cluster distance of data object x is defined as its contribution to the inner-cluster distance of the cluster it belong to: $D(x) = \sum_{i=1}^{|\mathbf{c}|} d(x, x_i)$ where $x \in \mathbf{c}$ and $x_i \in \mathbf{c}$. If the inner-cluster distance of a dataset $\mathbf{X}$ is calculated, it produces a set containing the inner-cluster distances of all data objects in the dataset: $D(\mathbf{X}) = \{D(\mathbf{x}_1), D(\mathbf{x_2}), \ldots, D(\mathbf{x_N}))$. The fitness function used to evaluate clustering $\mathbf{C}$ is the cumulative inner-cluster distance of the clustering:

$$f(\mathbf{C}) = \sum_{i=1}^{k} D(\mathbf{c}_i) \quad (2)$$

where $\mathbf{c}_i \in \mathbf{C}$.

3. ALGORITHM DESCRIPTION

The algorithm, called Heuristic Memetic Clustering Algorithm (HMCA) then works as follows: After an initial population of clusterings (individuals) is created and their fitness evaluated, HMCA iteratively approximates an optimal clustering. Each iteration (generation) a single local-search operator is applied to each (parent) individual, creating one (offspring or child) individual by moving one data object from one cluster to another (relabelling) guided by up to three heuristics. The offspring individual is then repaired (if necessary), and its fitness evaluated. Parent and child populations are then merged and made available for the next generation by a survivor selection operator. HMCA stop iterating based on a halting condition. Details of these operations and operators are provided below.

3.1. Fitness Evaluation Function

The fitness evaluation function used by HMCA evaluates individuals by calculating the cumulative inner-cluster distance of its clustering, defined in Equation 2. Calculating the inner-cluster distance of a cluster $\mathbf{c}$ requires $\frac{1}{2}|\mathbf{c}|(|\mathbf{c}| - 1)$ distance calculations, but all distance calculations required are between data objects in the dataset only. Speed up during experimentation is achieved by maintaining a cache of the distances between these data objects. Only required distances are calculated or taken from the cache, and flags in the individual are used to avoid recalculating data objects that have not been relabelled.

3.2. Individual Representation

Clusterings, i.e., individuals in HMCA are represented using an *Integer Label-based Encoding*, also used in [11, 12, 13]. This encoding uses a vector or array of length N, with each position assigned an integer label between 1 and k, so that the i-th position (gene) represents the i-th data objects with the i-th label indicating cluster $\mathbf{c}_i$ to which it is assigned. The fitness evaluation value is also stored by the individual, along with a number of convenience statistics and flags to reduce computation effort by the algorithm (see 3.1).

3.3. Initialisation, Repair, Survivor Selection, and Halting Condition

Individuals are initialised in two steps: first, all k clusters are assigned a unique uniform randomly selected data objects; then all remaining data objects are assigned a uniform randomly selected cluster. This initialisation method assures that initialised individuals do not contain empty clusters.

Empty clusters may be created during the search. A repair operator is applied to each newly generated individual to repair this. Upon finding an empty cluster, this repair operator assigns a uniform randomly selected data object from the largest (in size) cluster in the individual to the empty cluster.

The survivor selection operator creates a merged parent and offspring population equal in size to the population size parameter. Both populations are first merged, then ordered by fitness value, with individuals from the merged population unable to make the cut discarded. Only unique individuals are considered, duplicate clusterings are discarded as well with uniqueness is determined based on cluster content, not label values.

The HMCA uses a halting condition operator to terminate the algorithm after a fixed maximum number of distance calculates has been reached. The maximum number of distance calculations is set as a parameter of the algorithm. The halting condition is evaluated at the beginning of an iteration or generation, so the actual number of distance calculations may vary between runs. Compared to a maximum number of iterations, using a maximum number of distance calculations provides a fair measure for comparing different variants of the HMCA, e.g., between one where the search operator uses few distance calculations and one that uses many. Although the HMCA caches distance calculations, cache lookups are still counted as full calculations for the halting condition.

3.4. Heuristic Local-Search Operator

The search operator generates an offspring clustering in two steps: the first step selects a data object $x_i \in \mathbf{X}$; the second step selects a new cluster to which the data object is transferred: $\mathbf{c}_j \in \mathbf{C}$ with $x_i \in \mathbf{c}_l$ and $j \neq l$. Selection of the data object in step one can be done in two ways: selecting a data object from the whole set of data objects $x \in X$, or, first selecting a cluster $\mathbf{c} \in \mathbf{C}$ and then selecting a data object from the cluster ($x \in \mathbf{c}$). Selection of the data object and the cluster is done by a heuristic, which is in essence a selector.

The search operator may then sequentially use three types of heuristics: a cluster heuristic $H^{\mathbf{C}} \rightarrow \mathbf{c}$, a data object heuristic $H^{\mathbf{X}} \rightarrow x$, and a (cluster) label heuristic $H^{\mathbf{c}} \rightarrow \mathbf{c}$. Several selectors S are defined for use in these heuristics, each selector defined to use some criterion for selecting one element from a set: $S^{\max}(\mathbf{Y}) \rightarrow y$ selects the maximum element $y \in Y$; $S^{\min}(\mathbf{Y}) \rightarrow y$ selects the minimum element $y \in Y$; $S^{\text{rnd}}(\mathbf{Y}) \rightarrow y$ selects a uniform random element $y \in Y$; and $S^{\text{rnd}}_{\mathbf{W}}$ selects an element $y \in Y$ proportionally random, with each element $y \in Y$ weighted by corresponding weight $w \in \mathbf{W}$ ($|\mathbf{Y}| = |\mathbf{W}|$) and the probability of selection proportional to the corresponding weight.

For all three types of heuristics we define a benchmark, lower-bound, or reference heuristic, called the *null-heuristic*: $H^{\mathbf{C}}_0$, $H^{\mathbf{X}}_0$, and $H^{\mathbf{c}}_0$. The cluster *null-heuristic* $H^{\mathbf{C}}_0$ does not select a cluster, instead it tells the subsequent data object heuristic to select a data object from the set of all data object ($x \in \mathbf{X}$) instead of the from the set of data objects contained in a single (selected) cluster ($x \in \mathbf{c}$ with $\mathbf{c} \in \mathbf{C}$). The data object *null-heuristic* either selects a data object uniform random: $H^{\mathbf{X}}_0 : S^{\text{rnd}}(X) \rightarrow x$, or uniform randomly from a previously selected cluster c: $H^{\mathbf{X}}_0 : S^{\text{rnd}}(c) \rightarrow x$. Note that, without loss of generality, the search operator does not have a (null-) heuristic combination that uniform randomly selects a cluster to then uniform randomly select a data objects from it: $H^{\mathbf{X}} : S^{\text{rnd}}(S^{\text{rnd}}(C)) \rightarrow x$. The label *null-heuristic* uniform randomly selects a new cluster from the clustering, excluding the cluster to which the previously selected data object $x \in \mathbf{X}$ belongs to: $H^{\mathbf{c}}_0 : S^{\text{rnd}}(\mathbf{C}')$ with $\mathbf{C}' = \mathbf{C} - c$ where $x \in c$.

Note that the selectors above all operate sequentially on sets of sets, e.g., the selector $S^{\min}(S^{\max}(\mathbf{C})) \rightarrow x$ first selects the 'maximum' data object $c \in C$, and then selects the data object in the subsequently created set of selected data objects. This sequential property is used to define the remaining heuristics the search operator uses, but requires the definition of further set-transforms: T. Set-transforms apply a given operator to all elements of the set, producing a transformed set of the same size as the original. Two set transforms are required: division $T^{\text{div}}_z(\mathbf{Y}) = \{\frac{y_1}{z}, \frac{y_2}{z}, \ldots, \frac{y_{|\mathbf{Y}|}}{z}\}$; average, a special case of division: $T^{\text{avg}}(\mathbf{Y}) = \{\frac{y_1}{z}, \frac{y_2}{z}, \ldots, \frac{y_{|\mathbf{Y}|}}{z}\}$ where $z = \frac{\sum_{i=1}^{|\mathbf{Y}|} y_i}{|\mathbf{Y}|}$. Finally, a single set-operator is required: the set-summation, which sums the values of a set to provide a single value: $\sum(\mathbf{Y}) = \sum_{i=1}^{|\mathbf{Y}|} y_i$ with $y_i \in \mathbf{Y}$. Naturally, for all selectors, transforms, and the operator holds that the set must contain elements on which the defined operations are valid.

Cluster and data objects heuristics are then based colloquially around maximising the fitness contribution (f.c.), while label heuristics are based around minimising distance to cluster (d.t.c.). These can be used for a direct selection, or

one based on proportional selection (prop.), and taken as is, as an average (avg), or as a total. The non-null heuristics of the search operator can then be defined as follows:

1. Cluster heuristics:
   (a) Largest f.c.: $H_1^{\mathbf{C}} : S^{\max}(D(\mathbf{C})) \rightarrow \mathbf{c}$.
   (b) Largest average f.c.: $H_2^{\mathbf{C}} : S^{\max}(T^{\text{avg}}(D(\mathbf{C})) \rightarrow \mathbf{c}$.
   (c) Proportional f.c.: $H_3^{\mathbf{C}} : S^{\text{rnd}}_{D(\mathbf{C})}(\mathbf{C})) \rightarrow \mathbf{c}$.
   (d) Proportional average f.c.: $H_4^{\mathbf{C}} : S^{\text{rnd}}_{T^{\text{avg}}(D(\mathbf{C}))}(\mathbf{C}) \rightarrow \mathbf{c}$.
2. Data object heuristics ($\mathbf{Y} = \{\mathbf{X} | \mathbf{c} \in \mathbf{C}\}$):
   (a) Largest f.c.: $H_1^{\mathbf{X}} : S^{\max}(D(\mathbf{Y})) \rightarrow x$.
   (b) Largest average f.c.: $H_2^{\mathbf{X}} : S^{\max}(T^{\text{avg}}(D(\mathbf{Y})) \rightarrow x$.
   (c) Proportional f.c.: $H_3^{\mathbf{X}} : S^{\text{rnd}}_{D(\mathbf{Y})}(\mathbf{Y})) \rightarrow x$.
   (d) Proportional average f.c.: $H_4^{\mathbf{X}} : S^{\text{rnd}}_{T^{\text{avg}}(D(\mathbf{Y}))}(\mathbf{Y}) \rightarrow x$.
3. Label heuristics for previously selected x and $\mathbf{Y} = \mathbf{C} - \mathbf{c}$ with $x \in \mathbf{c}$:
   (a) Least d.t.c.: $H_1^{\mathbf{c}} : S^{\min}(S^{\max}_{\forall \mathbf{y} \in \mathbf{Y}}(d(x, \mathbf{y}))) \rightarrow \mathbf{c}$.
   (b) Lst avg d.t.c.: $H_2^{\mathbf{c}} : S^{\min}(T^{\text{div}}_{|\mathbf{y}|}(S^{\max}_{\forall \mathbf{y} \in \mathbf{Y}}(d(x, \mathbf{y})))) \rightarrow \mathbf{c}$.
   (c) Least total d.t.c.: $H_3^{\mathbf{c}} : S^{\min}(\sum_{\forall \mathbf{y} \in \mathbf{Y}}(d(x, \mathbf{y}))) \rightarrow \mathbf{c}$.
   (d) Prop. min. d.t.c.: $H_4^{\mathbf{c}} : S^{\text{rnd}}_{\frac{1}{S^{\max}_{\forall \mathbf{y} \in \mathbf{Y}}(d(x,\mathbf{y}))}}(\mathbf{Y})) \rightarrow \mathbf{c}$
   (e) Pp. min. avg d.t.c.: $H_5^{\mathbf{c}} : S^{\text{rnd}}_{\frac{1}{T^{\text{div}}_{|\mathbf{y}|}(S^{\max}_{\forall \mathbf{y} \in \mathbf{Y}}(d(x,\mathbf{y})))}}(\mathbf{Y})) \rightarrow \mathbf{c}$
   (f) Prop. min. total d.t.c.: $H_6^{\mathbf{c}} : S^{\text{rnd}}_{\frac{1}{\sum_{\forall \mathbf{y} \in \mathbf{Y}}(d(x,\mathbf{y}))}}(\mathbf{Y})) \rightarrow \mathbf{c}$

4. DATASETS

Three commonly used datasets were used to evaluate the performance of our algorithms. This facilitates the comparison of the performance of HMCA with other clustering algorithms. The first two datasets are collected data, the final dataset is generated.

The first dataset used is the Iris flower dataset introduced by Ronald Fisher [14]. Also called the Fisher's Iris dataset, it is a multivariate dataset of collected Iris flowers of $k = 3$ related species. The dataset consists of 50 samples from each of three species of Iris (Iris Setosa, Iris Virginica, and Iris Versicolor) for a total of $N = 150$ with $n = 4$ measured features for each sample.

The second dataset used is the Glass Identification dataset, or Glass dataset. It too is a multivariate dataset of $n = 9$ collected chemical compositions of glass. The dataset consists of $N = 217$ samples, clustered into $k = 6$ clusters of variable size, the largest includes 76 data objects, the smallest only 9.

The third and final dataset used is a generated Quadrature Amplitude Modulation (QAM) dataset. QAM is both an analog and a digital modulation scheme in which two analog or digital streams are modulated by the amplitudes of two carrier waves. The two carrier waves, usually sinusoids, are out of phase with each other by 90 degrees, producing a quadrature component, hence the name of the scheme. QAM is used extensively as a modulation scheme for digital telecommunication systems. Here 16-QAM is used where signal streams are originally modulated on a $4 \times 4 = 16$ grid, after which $12dB$ of Gaussian noise is added. $N = 1024$ data objects are uniform randomly generated, each with $n = 2$ features or coordinates. These are to be clustered in $k = 16$ clusters.

5. EXPERIMENTAL METHOD

Combining null- and non-null heuristics there are 5 cluster heuristics, 5 data object heuristics, and 7 label heuristics, for a total of $5 \cdot 5 \cdot 7 = 175$ variants of the HMCA. The experimental method for evaluating HMCA consists of running all 175 variants of the algorithm for different population size parameter settings. The number of clusters is fixed by parameter equal to the number of clusters of each dataset: Iris $k = 3$, Glass $k = 6$, and 16-QAM $k = 16$. The population size parameter was taken from the set $P_{\text{size}} \in \{1, 2, 5, 10\}$ so as to cover a range of possibilities. For the halting condition we set a maximum of $10,000,000$ distance calculations for the Iris and Glass datasets, and $20,000,000$ for the 16-QAM datasets. These values were found experimentally and allow the HMCA search to stabilise around a global optimum. Since HMCA is a stochastic problem solver, and thus non-deterministic, we ran HMCA 25 times and averaged all results. For each dataset we ran $25 \cdot 4 \cdot 175 = 17,500$ independent runs.

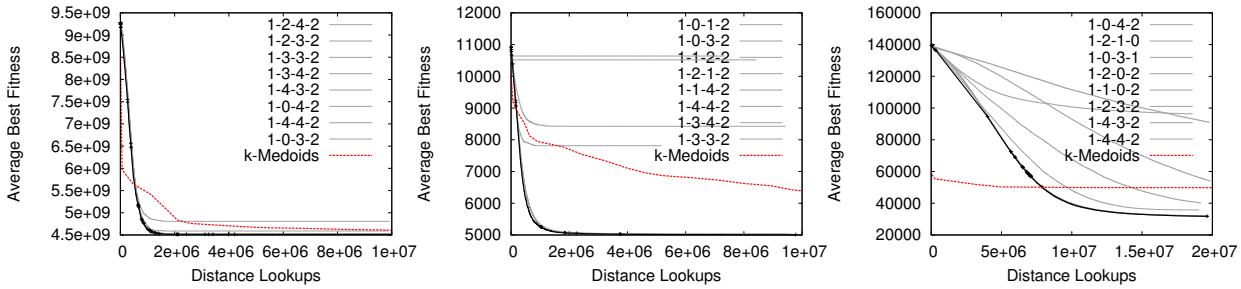
We evaluate the efficiency and effectiveness of HMCA by looking at the fitness value behaviour of the algorithm during the run, and by calculating the accuracy of the evolved clusterings. Clustering accuracy is calculated by compiling the *confusion matrix* of all best evolved clusterings. We then take the maximum sized cluster in the matrix for each generated clustering for each truth data value taken from the original dataset (for 16-QAM this is the dataset before noise was applied). All data objects labelled outside the maximum sized cluster are considered to be labelled incorrectly.

We compare the performance of the HMCA with the performance of the k-Medoids algorithm. Our version of the k-Medoids algorithm was reimplemented to make use of the same benefits as the HMCA (e.g. the cache) and restricted to the same limitations, i.e., the maximum number of distance calculations allowed. This ensures as fair a comparison as possible.

We believe that this exhaustive experimental setup, and these performance measures will provide us with the statistically significant results required to draw robust conclusions on both the efficiency and effectiveness of the algorithm. All relevant data is provided to facilitate comparison with other clustering algorithms.

6. EXPERIMENTAL RESULTS

Figure 1 shows the behaviour of the HMCA during the run. Along the horizontal axis the number of distance calculations are shown. Along the vertical axis the average values are shown. A selection of 8 better performing variants of the



a. Iris *b*. Glass *c*. 16-QAM

Fig. 1. HMCA Fitness value behaviour graphs for the Iris, Glass, and 16-QAM dataset.

HMCA are plotted as thin gray lines, each identified by a combination of 4 hyphenated numbers sequentially identifying: population size, cluster heuristic, data object heuristic, and label heuristic. The other variants, with poor performance, are not plotted so as to not overwhelm the graphs. The behaviour of the k-Medoids algorithm (using the same HMCA fitness evaluation function) is shown as a thick gray line (distance calculations for fitness evaluation were not counted for the halting condition). Finally, the thick black line is a compound line made up of the best performing variants generated by comparing interpolated values of *all* HMCA variants. A selection of the 4 best variants from the 8 shown is used for further analysis below. Subgraph a shows the results for the Iris dataset, b for the Glass dataset, and c for the 16-QAM dataset.

The graphs in Figure 1 show that for small number of clusters (Iris dataset, $k = 3$), k-Medoids has similar performance to HMCA, although HMCA gets to good clusterings quicker, i.e., using significantly less distance calculations. For larger k, HMCA significantly outperforms k-Medoids in clustering quality, even though k-Medoids produces significantly better initial clusterings than the two-step uniform random initialisation operator does for HMCA. This behaviour is the same for all three datasets: HMCA first needs to expend significantly more distance calculations to evolve a clustering of equal quality to k-Medoid generated initial clusters. After this, especially for larger k, HMCA quickly outpaces k-Medoids to find clusters of significantly better quality, using fewer distance calculations. For the 16-QAM dataset, k-Medoids requires a very large number of distance calculations improving its initial clustering.

Tables 1, 2, and 3 show the clustering accuracy of the HMCA on each of the three datasets for the 4 selected variants. Column 'variant' indicates the HMCA variant, column 'correct' the number of correctly clustered data objects cumulative for all 25 independent experiments for the final evolved individual with the best (lowest) fitness value. Column 'total' shows the total number of data objects ($25N$) for each dataset, with column 'acc.' showing accuracy of the clustering as the ratio between of the preceding two columns, higher is better. Values for the k-Medoids algorithm are included as well.

Table 1 shows the HMCA clustering accuracy on the Iris dataset, table 2 for the Glass dataset, and 3 for the 16-QAM dataset. For the Iris dataset there is no accuracy difference between the different variants of HMCA. All variants cluster $3,100$ from $3,750$ data objects correctly, for an accuracy of 0.83 (rounded to two decimals). For the Glass dataset 3 variants cluster data objects correctly with an accuracy of 0.53, with only variant 1-1-4-2 evolving a clustering with a significantly lower accuracy at 0.39. This is recognisable in Figure 1 b, variant 1-1-4-2 is the lighter thin line levelling out well before and above the other variants. For the 16-QAM dataset, again, 3 of the selected variants cluster the data objects with comparable accuracy; 0.81 or 0.82. Only variant 1-2-3-2 clusters the data objects with a lower accuracy of 0.79, recognisable in Figure 1 c as well. The values for k-Medoids correspond to the trends visible in Figure 1.

Comparison of the performance of HMCA with other algorithms proposed in literature (e.g. [9, 15]) is made more difficult since it is only possible on the (collected) Iris and Glass datasets, and many publications limit themselves to evaluating efficiency to the platform dependent wallclock-time-to-solution performance measure. We therefore limited comparison to comparing effectiveness, i.e., clustering quality. We found that the HMCA produced clusterings of similar or better quality to most proposed algorithms. Significant clustering quality improvement using the chosen distance calculation measure seem, at any rate, unlikely, while HMCA's performance gain is primarily focused on efficiency.

Overall, the parameters of the best performing variants of HMCA show best performance with only a single individual. Larger populations is not worth the maintenance cost required in relation to its performance. It is difficult to identify a best cluster heuristic, although there is a slight preference for the proportional random cluster heuristic. For data object heuristics the choice is more limited, with only the two proportional random data object heuristics used by the best performing HMCA variants. Only the least average dis-

variant	correct	total	acc.
k-Medoids	3112	3750	0.83
1-0-4-2	3100	3750	0.83
1-3-4-2	3100	3750	0.83
1-4-3-2	3100	3750	0.83
1-4-4-2	3100	3750	0.83

Table 1. Iris Accuracy Table

variant	correct	total	acc.
k-Medoids	2606	5350	0.49
1-3-3-2	2848	5350	0.53
1-3-4-2	2826	5350	0.53
1-4-4-2	2861	5350	0.53
1-1-4-2	2082	5350	0.39

Table 2. Glass Accuracy Table

variant	correct	total	acc.
k-Medoids	17837	25600	0.70
1-0-4-2	20612	25600	0.81
1-2-3-2	19565	25600	0.76
1-4-3-2	20906	25600	0.82
1-4-4-2	20964	25600	0.82

Table 3. 16-QAM Accuracy Table

tance to cluster label heuristic is used by the best performing HMCA variants. In general, the analysis of the experimental results indicate that choosing the heuristics, and the level of their determinism, becomes more important in the following order: cluster, data object, and (most important/deterministic) label heuristic. The best performance trend is then to select the cluster and/or data object proportionally random, with the label selected deterministically. In other words; it is more important to deterministically relabel a selected data object correctly than selecting which data object to relabel.

7. CONCLUSION

In this paper we introduce a novel clustering algorithm based on the Memetic Algorithm meta-heuristic, called HMCA. HMCA iteratively evolves clusterings by applying a single local-search operator. The novel feature of HMCA is that the local-search operator incorporates up to three types of selection heuristics. Several heuristics are defined for each type of selection, and combinations these heuristics constitute variants of HMCA. All variants were exhaustively tested on three commonly used benchmark problems and evaluated on both behaviour and quality of the found clusterings. HMCA was compared to the k-Medoids algorithm, reimplemented to provide a fair comparison. The results show that HMCA consistently provides better clustering solutions with significantly less computational effort.

8. REFERENCES

[1] L. Kaufman and P.J. Rousseeuw, "Finding groups in data – an introduction to cluster analysis," *Wiley Series in Probability and Mathematical Statistics*, 1990.

[2] J.P. Bigus, *Datamining with Neural Networks: solving business problems–from application development to decision support*, McGraw-Hill, 1996.

[3] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[4] G. Mecca, S. Raunich, and A. Pappalardo, "A new algorithm for clustering search results," *Data and Knowledge Engineering*, vol. 62, pp. 504–522, 2007.

[5] P. Baldi and S. Brunak, *Bioinformatics – The Machine Learning Approach*, MIT Press, 2nd ed. edition, 2001.

[6] A.K. Jain, M.N. Mutry, and P.J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, pp. 264–323, 1999.

[7] E. Falkenauer, *Genetic Algorithms and Grouping Problems*, John Wiley & Sons, 1998.

[8] V.J. Rayward-Smith, "Metaheuristics for clustering in KDD," in *In Proc. IEEE Congress on Evolutionary Computation*, 2005, pp. 2380–2387.

[9] E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas, and A.C.P.L.F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Trans. on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 39, no. 2, pp. 133–155, 2009.

[10] B.G.W. Craenen and A.E. Eiben, "Hybrid evolutionary algorithms for constraint satisfaction problems: Memetic overkill?," in *Proceedings of the 2005 Conference on Evolutionary Computing (CEC 2005)*. Sep 2005, IEEE Computer Society Press.

[11] K. Krishna and N. Murty, "Genetic k-means algorithm," *IEEE Trans. on Systems, Man, and Cybernetics – Pt. B: Cybernetics*, vol. 29, pp. 433–439, 1999.

[12] R. Krovi, "Genetic algorithms for clustering: A preliminary investigation," in *Proceedings of the 25th Hawaii Int. Conference on System Sciences*, 1992, vol. 4, pp. 540–544.

[13] C.A. Murthy and N. Chowdhury, "In search of optimal clusters using genetic algorithms," *Pattern Recognition Letters*, pp. 825–832, 1996.

[14] R.A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[15] J. Eggermont, J.N. Kok, and W.A. Kosters, "Genetic programming for data classification: partitioning the search space," in *SAC*, 2004, p. 1001.