Genetic Algorithm for Finding Cluster Hierarchies

Christian Böhm, Annahita Oswald, Christian Richter,
Bianca Wackersreuther, and Peter Wackersreuther

Ludwig-Maximilians-University,
Department for Informatics
Oettingenstr. 67
80538 Munich, Germany
{boehm,oswald,wackersb,wackersr}@dbs.ifi.lmu.de
richterch@cip.ifi.lmu.de

Abstract. Hierarchical clustering algorithms have been studied extensively in the last years. However, existing approaches for hierarchical clustering suffer from several drawbacks. The representation of the results is often hard to interpret even for large datasets. Many approaches are not robust to noise objects or overcome these limitation only by difficult parameter settings. As many approaches heavily depend on their initialization, the resulting hierarchical clustering get stuck in a local optimum. In this paper, we propose the novel genetic-based hierarchical clustering algorithm **GACH** (**G**enetic **A**lgorithm for finding **C**luster **H**ierarchies) that solves those problems by a beneficial combination of genetic algorithms, information theory and model-based clustering. GACH is capable to find the correct number of model parameters using the Minimum Description Length (MDL) principle and does not depend on the initialization by the use of a population-based stochastic search which ensures a thorough exploration of the search space. Moreover, outliers are handled as they are assigned to appropriate inner nodes of the hierarchy or even to the root. An extensive evaluation of GACH on synthetic as well as on real data demonstrates the superiority of our algorithm over several existing approaches.

1 Introduction

A genetic algorithm (GA) is a stochastic optimization technique based on the mechanism of natural selection and genetics, originally proposed by [15]. The general idea behind a GA is that the candidate solutions to an optimization problem (called *individuals*) are often encoded as binary strings (called *chromosomes*). A collection of these chromosomes forms a *population*. The evolution initially starts from a random population that represents different individuals in the search space. In each generation, the fitness of every individual is evaluated, and multiple individuals are then selected from the current population based on Darwin's principle "Surviving of the fittest". These individuals build the mating pool for the next generation. The new population is then formed by the application of recombination operations like *crossover* and *mutation*. A GA commonly terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. An excellent survey of GAs along with the programming structure used can be found in [10].

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 349–363, 2011.

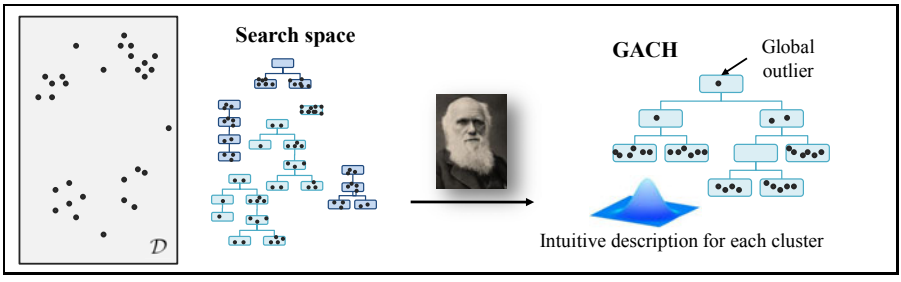


Fig. 1. The search space of the solutions for the hierarchical clustering problem is extremely large. Genetic algorithms help to determine the global optimum. GACH is a genetic algorithm for finding the most meaningful cluster hierarchy. Outliers are assigned to appropriate inner nodes. Each cluster content is described by an intuitive description in form of a PDF.

GAs have been successfully applied to a variety of challenging optimization problems, like image processing, neural networks and machine learning, etc. [21,26,2]. Solving the NP-hard clustering problem makes GA therefore a natural choice such as in [17,24,6,18,23]. The clustering research community focuses on clustering methods where the grouped objects are described by an intuitive model of the data [7] or clustering methods that are particularly insensitive to outliers [9]. Moreover, several approaches have also addressed the question, how to avoid difficult parameter settings such as the number of clusters, e.g. [22,14,3,4]. Most of them meet this question by relating the clustering problem with the idea of data compression.

Here we present a novel genetic algorithm for finding cluster hierarchies, called GACH. We use an information-theoretic fitness function to effectively cluster data into meaningful hierarchical structures without requiring the number of clusters as an input parameter. Our major contributions are:

- *Fitness*: The fitness of different chromosomes is optimized using an optimization technique that is based on the Minimum Description Length (MDL) principle.
- *No difficult parameter-setting*: Besides the parameters that are specific for a GA, GACH requires no expertise about the data (e.g. the number of clusters).
- *Flexibility*: By the use of a GA-based stochastic search GACH thoroughly explores the search space and is therefore flexible enough to find the correct hierarchical cluster structure and is insensitive to the initialization.
- *Outlier-robust*: Outliers are assigned to the root of the cluster hierarchy or to an appropriate inner node, depending on the degree of outlierness.
- *Model description*: The content of each cluster is described by a PDF.

The rest of the paper is structured as follows: Section 2 reviews some well-known approaches in the field of hierarchical and information-theoretic clustering including several genetic algorithms. In Section 3 we face the general idea behind genetic algorithms and present our proposed method GACH which searches for optimal hierarchical clustering results regarding accuracy. An extensive evaluation of GACH including method comparison and benchmarking of synthetic as well as real data is provided in Section 4 followed by the conclusion of the paper in Section 5.

2 Related Work

Although a huge amount of clustering approaches are available today, almost all of them suffer from at least one of the following drawbacks: they are restricted to partitioning clustering, and/or they are sensitive to outliers, and/or they need user-defined parameters (e.g. the number of clusters). In this section, we survey the work on hierarchical and model-based clustering and summarize important beneficial results from information theory in the field of clustering. Finally, we focus on GAs that were designed for solving the optimization problem of finding a meaningful clustering.

Hierarchical Clustering. A widespread approach to hierarchical clustering is Single Link [16]. It produces a graphical output, the so-called *dendrogram*. Cuts through the dendrogram at various levels obtain partitioning clusterings. However, for complex datasets it is hard to define appropriate splitting levels, which correspond to meaningful clusterings. Furthermore, outliers may cause the well-known Single Link effect. Also, for large datasets, the fine scale visualization is not appropriate. OPTICS [1] avoids the Single Link effect by requiring a minimum object density for clustering, i.e. $MinPts$ number of objects are within a hyper-sphere with radius ϵ. Additionally, it provides a more suitable visualization, the so-called *reachability plot*. However, the right choice of the parameters is not intuitive and has significant impact on the performance of the algorithm and the accuracy of the results. Furthermore, the problem that only certain cuts represent useful clusterings still remains unsolved.

Model-based Clustering. Model-based clustering assumes that the data is generated by a finite mixture of underlying probability distributions such as multivariate normal distributions. A commonly used algorithm for model-based clustering is the Expectation-Maximization (EM) algorithm [7]. After a suitable initialization, EM iteratively optimizes a mixture model of k Gaussians until no further significant improvement of the log-likelihood of the data can be achieved. Two common problems of EM are (1) the algorithm may get stuck in a local optimum and (2) the quality of the result strongly depends on an appropriate choice of k. Besides the classical EM, multiple hierarchical extension can be found in the literature [25,5,11]. However, each of these approaches needs a suitable parameter setting for the number of hierarchy levels.

Information-theoretic Clustering. Difficult parameter settings are often avoided by information-theoretic clustering. X-Means [22], G-Means [14] and RIC [3] try to find the optimal k in partitioning clustering by balancing data likelihood and model complexity. This sensitive trade-off is rated by model selection criteria, e.g. Minimum Description Length (MDL) [12]. The RIC algorithm uses MDL to allow for defining a coding scheme for outliers and to identify non-Gaussian clusters. However, these methods are not hierarchical but only partitioning methods. An EM-like algorithm for information-theoretic hierarchical clustering, called ITCH, is presented in [4]. After intitalization ITCH rearranges the hierarchy in a Greedy-like search which often converges only to a local optimum.

Genetic Clustering Algorithms. A genetic k-means algorithm was introduced by Krishna and Murty [17]. Scheunders [24] published a genetic variant of the c-means clustering algorithm. Some kind of semi-supervised genetic clustering was presented by [6],

which is also a k-means based approach. In [18] the authors try to improve a fitness function concerning the space restrictions on the one hand and the building blocks on the other hand. One recent approach is the one by Pernkopf and Bouchaffra [23], which combines the benefits of a GA with model-based clustering to find a nearly optimal solution for a given number of clusters. With the help of a MDL criterion the correct number of clusters is determined fully automatically. All these methods are only applicable to partitioning clustering or suffer from the problem that human interaction is still necessary to enter a suitable k for the number of clusters resulting from a fixed length of chromosomes. The detection of noise and outliers is not supported at all.

3 GACH – Genetic Algorithm for Finding Cluster Hierarchies

In this section, we describe the basic components of a GA and introduce necessary modifications to use a GA on cluster hierarchies. Finally, we present GACH as an algorithmic combination of all this components.

3.1 Chromosomal Representation of Cluster Hierarchies

Each chromosome specifies one solution to a defined problem. For GACH, a chromosome is the encoding of a hierarchical cluster structure (HCS), that has to address the three following features:

- Storage of k clusters, where k is an arbitrary number of clusters.
- Representation of the hierarchical relationship between clusters forming a tree $\mathcal{T}$ of clusters.
- Encoding of the cluster representatives, i.e. the parameters of the underlying PDF. For GACH, we represent each cluster by a Gaussian PDF. Note that our model can be extended to a variety of other PDFs, e.g. uniform or Laplacian.

With these requirements a chromosomal representation of a HCS is defined as follows:

Definition 1 (Chromosomal HCS)
*(1) A **chromosomal HCS** HCS_{Chrom} is a dynamic list storing k cluster objects.*
(2) Each cluster C holds references to its parent cluster and to its subclusters. Besides that, the level l_C for a cluster C denotes the height of the descendant subtree. This implies that the root has the highest level and the leaves have level 0.
(3) The parameters of the underlying Gaussian PDF of cluster C, the mean value μ_C and σ_C, are modeled as additional parameters of the cluster object C.
(4) Each cluster C is associated with a weight W_C, where $\sum_{i=0}^{k-1} W_{C_i} = 1$.

The underlying PDF of a cluster C is a multivariate Gaussian in a d-dimensional data space which is defined by the parameters μ_C and σ_C (where μ_C and σ_C are vectors from a d-dimensional space) by the following formula:

$$N(\mu_C, \sigma_C, x) = \prod_{1 \le i \le d} \frac{1}{\sqrt{2\pi\sigma_{C,i}^2}} \cdot \mathbf{e}^{-\frac{(x_i - \mu_{C,i})^2}{2\sigma_{C,i}^2}}$$

GACH assigns each point x in a dataset $\mathcal{D}$ *directly* to that cluster $C \in HCS_{Chrom}$ the probability density of which is maximal at the position of x:

$$C(x) = \arg \max_{C \in HCS_{Chrom}} \{W_C \cdot N(\mu_C, \sigma_C, x)\}.$$

3.2 Initialization of GACH

Basically the initial set of a population consists of a randomly generated set of individuals. This strategy is also processed by GACH, where in a first step a random number $\tilde{k}$ of clusters is selected for each structure HCS_{Chrom}. Then a simple k-means algorithm divides the dataset into $\tilde{k}$ clusters that act as the leafs of the initial hierarchy. Finally, these clusters are combined by one additional root cluster. Hence, the initialization process results in a 2-level hierarchy that consists of $\tilde{k} + 1$ nodes. Each cluster C is described by random parameters and is associated to a weight $W_C = \frac{1}{k}$.

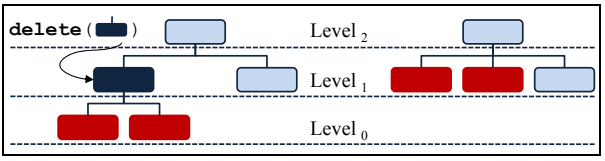
3.3 Reproduction

In order to generate the next population of cluster hierarchies GACH uses several genetic operators: mutations (`delete`, `add`, `demote` and `promote`) and `crossover`. We define these operators particularly for the hierarchical clustering problem here.

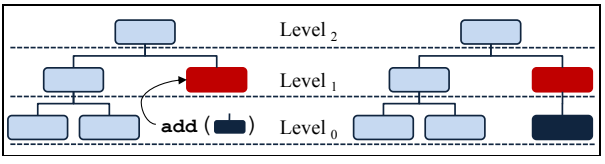
The `delete` operator deletes a specific cluster C (except the root) with a deletion rate p_{del} from the HCS. This results in structure HCS′ that does not contain the cluster C any more. This proceeding is illustrated in Figure 2(a). Here, the cluster C is marked by dark blue color. The level of each direct and undirect subcluster of C (marked in red) is decreased by 1. The former parent node of C, the root node in our example, becomes the parent node of all direct subclusters of C.

The operator `add` adds direct subclusters to an arbitrary cluster C of the hierarchy with an add rate p_{add} (normally $p_{del} = p_{add}$). The number of added subclusters is bounded by an upper limit value max_{new}. Figure 2(b) illustrates an example for the application of the `add` operator to a HCS. One subcluster (marked in dark blue color) is added to the red cluster. Since the cluster content of a added subcluster C_{add} is covered by the cluster content of C, we calculate random parameters based on μ_C and σ_C. In particular, we add a random factor r to both parameters, where r is a vector from a d-dimensional space: $\mu_{C_{add}} = \mu_C + r \qquad \sigma_{C_{add}} = \sigma_C + r$.

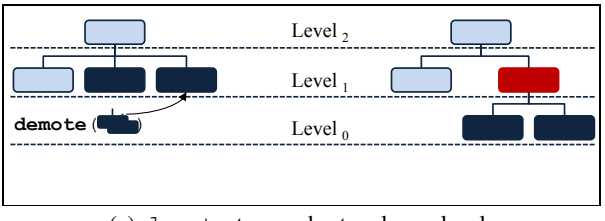
The third mutation operator is the `demote` operator, that can be motivated as follows: Assume a dataset consisting of three clusters C_1, C_2 and C_3, where C_1 holds a large number of objects, clusters C_2 and C_3 are smaller ones but they are locally close to each other. Then one could create a HCS with one root node and C_1, C_2 and C_3 as direct subclusters (cf. Figure 2(c)) which provides only a very coarse view of the dataset. But, if we combine the two smaller clusters (marked in dark blue) and demote them with a demote rate p_{dem} to a lower level with a common parent cluster (marked in dark red), we are able to get a more detailed look on our data. The parameters of the inserted cluster are obtained by the average of the parameters of the combined clusters. Note that demoting only one cluster is equal with the `add` operator. Hence, we apply `demote` on at least two clusters.



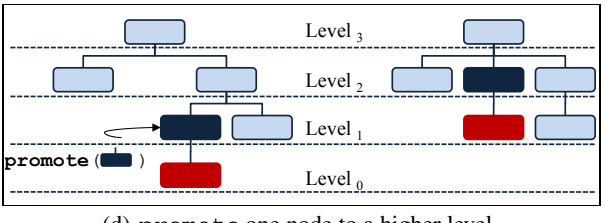
(a) `delete` one node of the hierarchy.



(b) `add` one node to the hierarchy.



(c) `demote` two nodes to a lower level.



(d) `promote` one node to a higher level.

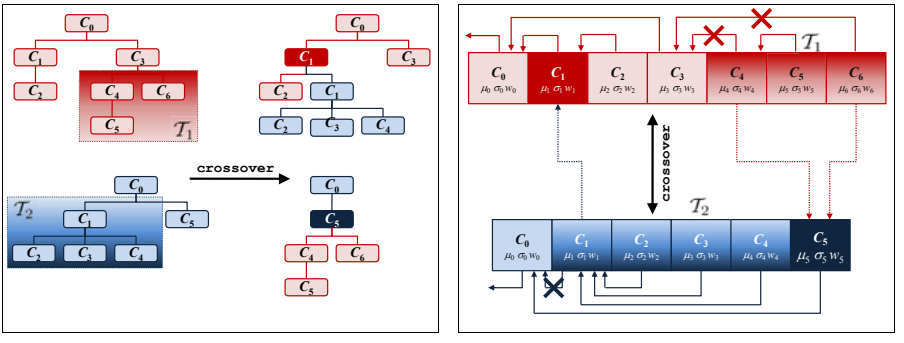
Fig. 2. Summarization of the mutation operators used for GACH

The `promote` operator lifts a cluster C from a lower level to the level right above with a promotion rate p_{pro}, if and only if C is at least two levels underneath the root cluster. Consequently all subclusters of C are lifted accordingly. In Figure 2(d) the dark blue cluster is promoted from level 1 to level 2. Hence also the red subcluster is lifted one level above. The parent of the parent node of the dark blue cluster (here the root node) becomes the parent node of C in the resulting hierarchy HCS′, together with the correct rearrangement of all subclusters.

The operator `crossover` exchanges information among two different structures. In general the information of two different chromosomes is combined in order to obtain a new individual with superior quality. GACH performs a crossover between two selected hierarchies HCS_1 and HCS_2 with a crossover rate p_{co} as follows:

1. Remove a selected subtree $\mathcal{T}_1$ entirely from HCS_1.
2. Remove a selected subtree $\mathcal{T}_2$ entirely from HCS_2.
3. Select a random node in HCS_1 and insert $\mathcal{T}_2$.
4. Select a random node in HCS_2 and insert $\mathcal{T}_1$.

Figure 3(a) illustrates this procedure exemplarily for two selected hierarchies. The subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ are removed from the red and the blue HCS respectively. $\mathcal{T}_1$ is then inserted into the blue HCS as subtree of the dark blue node. Analogously $\mathcal{T}_2$ is inserted as subtree of the dark red cluster in the red HCS. Figure 3(b) describes the same procedure w.r.t. a chromosomal representation of both hierarchies. For simplicity, only the pointers to the parent cluster are considered.



(a) Hierarchical representation. (b) Chromosomal representation.

Fig. 3. The `crossover` operator for two selected hierarchies. The subtree $\mathcal{T}_1$ of the red hierarchy is exchanged with the subtree $\mathcal{T}_2$ of the blue hierarchy, visualized by a hierarchical (3(a)) and a chromosomal representation (3(b)).

3.4 Fitness Function

Following the Darwin's principle "Survival of the fittest" naturally only individuals with highest fitness can survive and those that are weaker become extinct. A GA adopts this aspect of evolution by the use of a fitness function. GACH uses the $hMDL$ criterion formalized in [4] which evaluates the fitness of a chromosomal HCS by relating the clustering problem to that of data compression by Huffman Coding. The authors define the coding scheme for a cluster hierarchy as follows:

$$hMDL_{HCS} = \sum_{C \in HCS} \left(cost(C) - nW_C \log_2(W_C) - \log_2(\sum_{x \sqsubseteq \text{parent of } C} W_x) \right)$$

The coding cost for each cluster $C \in HCS$ is determined separately and summed up to the overall coding cost of the complete HCS. Points that are directly assigned to the cluster C together with the parameters μ_C and σ_C of the underlying Gausssian PDF are coded by $cost(C)$. The point to cluster assignment is coded by the so-called ID cost of each data point $x \in C$ and are given by $-nW_C \log_2(W_C)$ where W_C is the weight of cluster C and n the number of points. The binary logarithm is used to represent the code length in bits. Clusters with higher weight are coded by a short code pattern whereas longer code patterns are assigned for smaller clusters with lower weight. The ID cost for the parameters are formalized by $-\log_2(\sum_{x \sqsubseteq \text{parent of } C} W_x)$ whereas constant ID cost are defined for the parameters of the root node.

The better the statistical model (the HCS) fits to the data the higher the compression rate, and thus the lower the coding cost are. Using this coding scheme as fitness function ensures the selection of that chromosome HCS_{Chrom} that fits best to the data.

3.5 Selection

The selection function chooses the individuals to form the offspring population out of a set of given individuals, according to their fitness. For GACH, we use the well-known weighted roulette wheel strategy [19]. Imagine that each HCS_{Chrom} represents a number on a roulette wheel, where the amount of numbers refers to the size of the population. In addition, we assign a weight to each number on the roulette wheel, depending on the fitness of the underlying chromosome. That means that the better the fitness of a chromosome, the higher its weight on the roulette wheel, i.e. the higher the chance to get selected for the offspring population. Note that there is the chance that one chromosome is selected multiple times. GACH forms a new population that has as much individuals as the former population.

3.6 Algorithmic Description

Now we are putting the pieces together to define the algorithmic procedure of GACH, summarized in Algorithm 1. An initial population is built as described in Section 3.2. This population is evaluated according to the fitness function introduced in Section 3.4 which means that GACH determines the coding cost for each cluster hierarchy of the population. In order to optimize the point to cluster assignment of each HCS and to provide an additional model of the data, we apply an hierarchical EM algorithm on each cluster structure, as suggested in [4].

Algorithm 1. GACH

1: $count_{pop} \leftarrow 0$
2: initialize $population(count_{pop})$
3: evaluate $population(count_{pop})$
4: **while** $(count_{pop} \leq pop_{max})$ **do**
5: $count_{pop} \leftarrow count_{pop} + 1$
6: select $population(count_{pop})$ from $population(count_{pop} - 1)$
7: reproduce $population(count_{pop})$
8: evaluate $population(count_{pop})$
9: **end while**

The population resulting from the initialization undergoes several mutation and crossover operations within pop_{max} number of generations in an iterative way. In each iteration the next population is selected according to the weighted roulette wheel strategy (cf. Section 3.5) and undergoes several reproduction procedures (cf. Section 3.3). Each operation is processed with a certain probability which is extensively evaluated in Section 4. After optimizing the point to cluster assignment, GACH determines the

fitness of each HCS_{Chrom}. The algorithm terminates if a specified maximum number of new populations pop_{max} is reached. The experiments show that the HCS can be optimized even with small generation sizes.

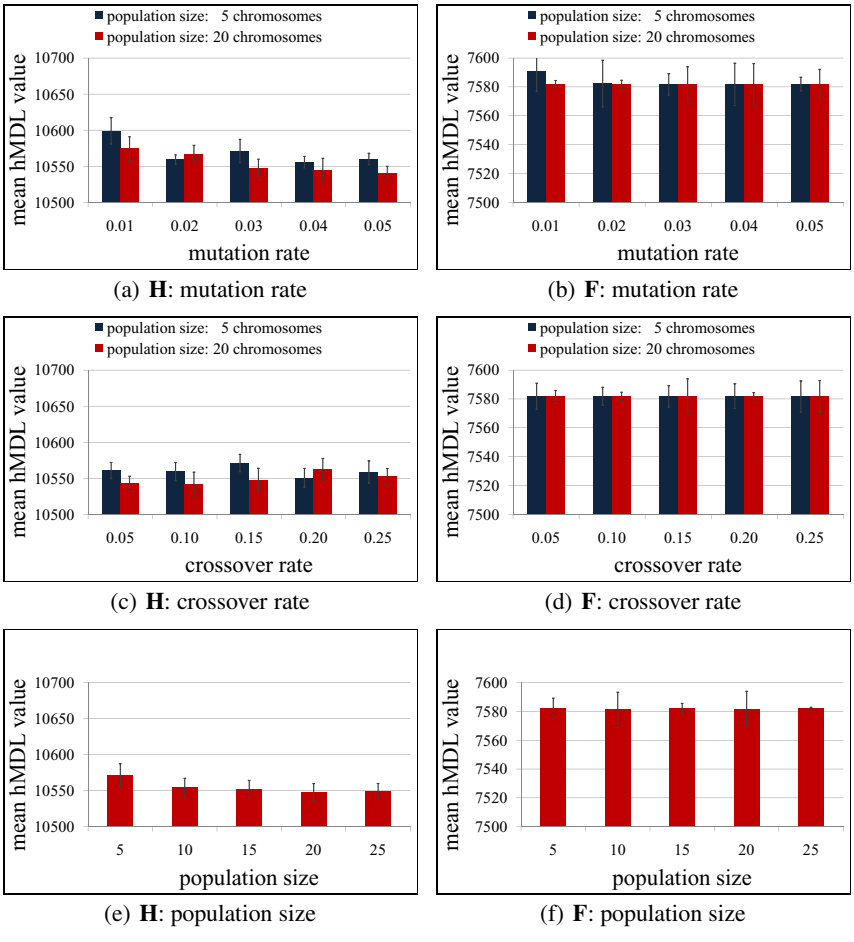
4 Experimental Evaluation

Now we demonstrate that the genetic parameters (mutation rate, crossover rate and population size) do not affect the effectiveness of GACH in a major way. Nevertheless, we provide a suitable parametrization that enables the user to receive good results independent of the used dataset. Based on this, we compared the performance of GACH to several representatives of various clustering paradigms on synthetic and real world data. We selected the most widespread approach to hierarchical clustering Single Link (SL) [16], the more outlier-robust hierarchical clustering algorithm OPTICS [1] (requiring $MinPts$ and ϵ), with optimal parameters w.r.t. accuracy. Furthermore, we chose RIC [3], an outlier-robust and information-theoretic clusterer, and finally ITCH [4] which is a recent EM-based hierarchical information-theoretic clustering approach, that suffers from the problem that the result often only represents a local optimum. As ITCH strongly depends on its initialization, we used the best out of ten runs in this case. For the SL experiments, we used the Matlab implementation. OPTICS was provided by WEKA [13]. For RIC and ITCH we used the original Java implementations by the authors.

4.1 Evaluation of Genetic Parameters

We applied GACH on two different datasets to evaluate the mutation and crossover rates and the impact of the population size on the quality of the results w.r.t. the fitness function, introduced in Section 3.4. One dataset consists of 1,360 (2d)-data points that form a true hierarchy of six clusters. The second dataset covers 850 (2d)-data points that are grouped in two flat clusters. For each experiment, we present the mean $hMDL$ value and the corresponding standard deviation over ten runs. GACH turned out to be very robust and determines very good clustering results ($Prec > 90\%$, $Rec > 90\%$) indepent of the parametrizations.

Different Mutation Rates. We evaluated different mutation rates ranging from 0.01 to 0.05 on two different population sizes and a fixed crossover rate of 0.15. As a mutation is performed by one of the four operations `delete`, `add`, `demote` or `promote` the mutation rate is the sum of p_{del}, p_{add}, p_{dem} and p_{pro} (cf. Section 3.3). As `demote` and `promote` turned out to be essential for the quality of the clustering results p_{dem} and p_{pro} are typically parametrized by a multiple of p_{del} or p_{add}. This is due to the fact that the optimal number of clusters which is influenced by p_{del} and p_{add} is determined very fast by the fitness function, but p_{dem} or p_{pro} have an impact on the hierarchical structure of the clusters that has to be adjusted during the run of GACH. Figures 4(a) and 4(b) demonstrate that the mutation rate has no outstanding effect on the clustering result, neither on a hierarchical nor on a flat dataset. Higher mutation rates result in higher runtimes (3,388 ms for mutation rate = 0.05 vs. 1,641 ms for mutation rate = 0.01



(a) **H**: mutation rate (b) **F**: mutation rate

(c) **H**: crossover rate (d) **F**: crossover rate

(e) **H**: population size (f) **F**: population size

Fig. 4. Mean fitness of resulting clusterings on (**H**)ierarchical and (**F**)lat datasets w.r.t. the genetic parameters mutation rate, crossover rate and population size

on hierarchical dataset, population size = 5). However, a higher mutation rate provides more flexibility. Hence, we achieved slightly better results with a mutation rate of 0.05 ($hMDL = 10,520$) compared to a mutation rate of 0.01 ($hMDL = 10,542$).

Different Crossover Rates. We compared different crossover rates p_{co} ranging from 0.05 to 0.25 in combination with a mutation rate of 0.05 on two different population sizes. Figures 4(c) and 4(d) show that the performance of GACH is almost stable w.r.t. the different parameterizations of p_{co}. Especially on a flat dataset, a higher p_{co} has no impact on the clustering result. GACH achieved a nearly optimal $hMDL$ value in almost every run, even for relatively small population sizes. Higher p_{co} values enable GACH to examine the search space more effectively as the crossover between two strong individuals produces an even fitter individual. Therefore, we need less generations to find good clustering results, e.g. the result of GACH on the hierarchical dataset

using five structures was determined after 75 generations (1,993 ms per generation) with $p_{co} = 0.05$, and after 61 generations (2,553 ms per generation) with $p_{co} = 0.25$.

Different Population Sizes. We tested the impact of the population size on the quality of the clustering result. We used populations that cover 5, 10, 15, 20 and 25 hierarchical cluster structures in combination with a mutation rate of 0.05 and a crossover rate of 0.15. Figures 4(e) and 4(f) show again the mean $hMDL$ value over ten runs for each population size on two different datasets. Especially the plot of the hierarchical dataset demonstrates that a higher population size tends to produce better results, which can be explained by the fact that a higher population size provides more variation opportunities whereby a global optimum can be reached easier. However, a large number of chromosomes cause a considerable amount of runtime. One generation covering five chromosomes took 2,462 ms on average, the computation of a generation consisting of 25 chromosomes took 9,229 ms. Hence, we use a population size consisting of ten cluster structures in combination with a mutation rate of 0.05 and a crossover rate of 0.15 in the following experiments.

4.2 Competitive Performance of GACH

For these experiments we use two different synthetic datasets DS_1 and DS_2. DS_1 is composed of 987 (2d)-data points that form a hierarchy of six clusters surrounded by local and global noise (cf. Figure 5(a)). DS_2 consists of 1,950 (2d)-data points that are grouped in three flat strongly overlapping clusters (cf. Figure 5(b)). For each dataset, the clustering results were assessed against a ground-truth classification. To compare the clustering results produced by different approaches, we chose measures that comprise fix bounds. More precisely, we selected the following recently proposed information-theoretic methods [20]: the Normalized Mutual Information (NMI), the Adjusted Mutual Information (AMI), which corrects the NMI for randomness. Both measures have value 1 if the resulting clustering corresponds exactly to the ground-truth classification, and 0 when both clusterings are totally independent. Furthermore, we chose the well-known Precision (Prec) and Recall (Rec) from information retrieval. Finally, we use the DOM [8] value that corresponds to the number of bits required to encode the class labels when the cluster labels are known, i.e. low DOM values represent good clustering results.

Evaluation w.r.t. Dataset $\mathbf{DS_1}$. These experiments demonstrate that GACH performs at least as good as the parameter dependent approach OPTICS ($MinPts = 6, \epsilon = 0.9$) and the recently proposed method ITCH, while being much more accurate than RIC and SL. The reachability plot of OPTICS is provided in Figure 5(c), the SL dendrogram is shown in Figure 5(e). Although DS_1 seems to be an easy to cluster dataset, SL and RIC fail in assigning the local and global outliers to the correct clusters. Both, GACH and ITCH were able to determine the right cluster structure (shown in Figure 5(g)). However, GACH outperformed ITCH w.r.t. accuracy, as GACH results in a different points to clusters assignment. Therefore, GACH shows the best performance on DS_1 concerning the information-theoretic measures NMI and AMI. Moreover, the values for Precision and Recall indicate that 94% of all data points are assigned to the true cluster

and 95% of the cluster contents were detected correctly by GACH. Finally, this result can be coded most efficiently as stated by the low DOM value of 0.4193. The evaluation on DS_1 is summarized in Table 1.

Table 1. Performance of GACH on DS_1

	GACH	ITCH	RIC	OPTICS	SL
NMI	**0.9346**	0.9265	0.8673	0.9045	0.9110
AMI	**0.9159**	0.8999	0.7678	0.8662	0.8997
PREC	0.9404	0.9222	0.6438	0.8626	**0.9555**
REC	**0.9514**	0.9422	0.7720	0.9200	0.9169
DOM	**0.4193**	0.4454	0.6638	0.4960	0.4765

Evaluation w.r.t. Dataset $\mathbf{DS_2}$. Neither OPTICS nor SL were able to detect the true cluster structure of DS_2. Both fail because of a massive Single Link effect and therefore the reachability plot provided by OPTICS (cf. Figure 5(d)) and the dendrogram produced by SL (cf. Figure 5(f)) do not uncover any cluster structure which leads to a clustering where all objects belong to the same cluster. Also RIC determines only one single cluster. ITCH separates the dataset into only two clusters. In contrast, GACH identifies all clusters in the dataset correctly. Hence, GACH turned out to be the only algorithm that handles datasets with strongly overlapping clusters successfully. It shows the best values w.r.t. information-theoretic criterions, while being very accurate. Its result causes only a coding cost of 0.3325 compared to more than 0.9 for almost all other approaches. The evaluation on DS_2 is summarized in Table 2.

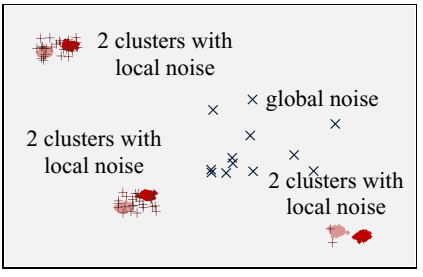
Table 2. Performance of GACH on DS_2

	GACH	ITCH	RIC	OPTICS	SL
NMI	**0.6698**	0.6316	0.0000	0.0000	0.0000
AMI	**0.5877**	0.4030	0.0000	0.0000	0.0000
PREC	**0.9184**	0.8227	0.2130	0.5016	0.6750
REC	**0.9226**	0.8913	0.4615	0.4615	0.4615
DOM	**0.3325**	0.4226	0.9184	0.9184	0.9184

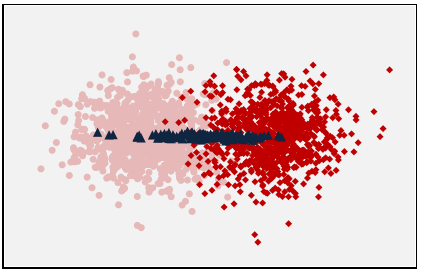
4.3 Application of GACH on Real World Data

We tested the practical application of GACH on several real world datasets. Due to space restrictions, we selected the high dimensional *Wine* dataset[1] for presentation. It contains 178 (13-d)-data objects resulting from a chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. Hence, a ground-truth classification structures the data into one root node covering the whole dataset and three subclusters defining the three cultivars. This structure was only determined by GACH resulting in high validity values (cf. Table 3). Most of the competitors did not even find

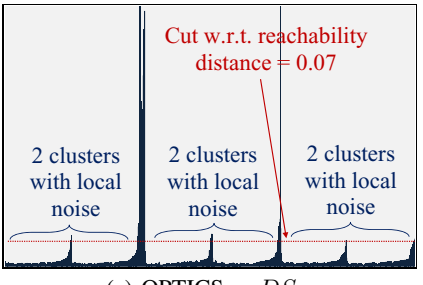
[1] `http://archive.ics.uci.edu/ml/datasets/Wine`



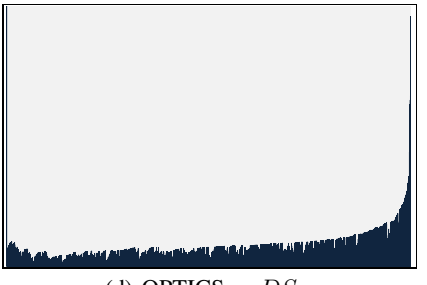
(a) Plotted DS_1.



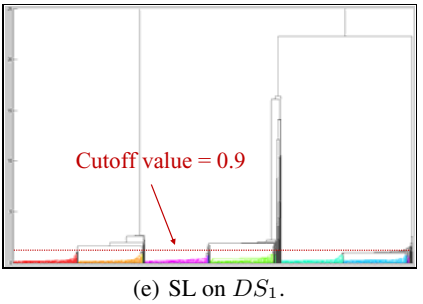
(b) Plotted DS_2.



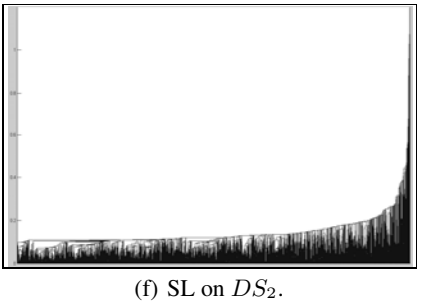
(c) OPTICS on DS_1.



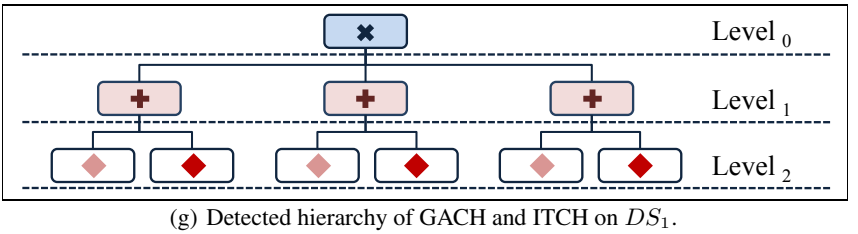
(d) OPTICS on DS_2.



(e) SL on DS_1.

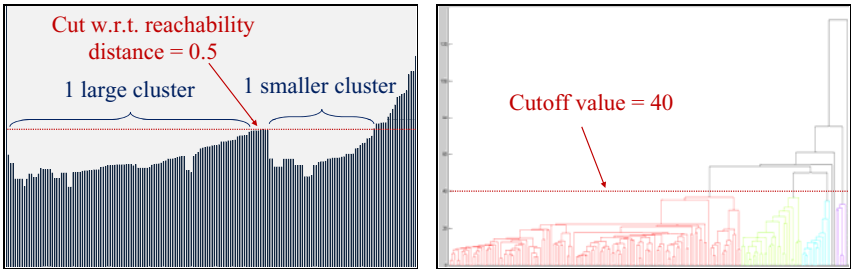


(f) SL on DS_2.



(g) Detected hierarchy of GACH and ITCH on DS_1.

Fig. 5. Competitive evaluation of GACH on two different synthetic datasets. DS_1 forms a hierarchy including local and global noise, DS_2 is a flat dataset of three overlapping clusters.



(a) OPTICS on *Wine*. (b) SL on *Wine*.

Fig. 6. Competitive evaluation of GACH on real-world data

the right number of clusters. For example, OPTICS uncovers two different clusters (cf. Figure 6(a)) and RIC even merges all data points in only one single cluster. SL detects four different clusters concerning a cutoff value of 40. Besides the *Wine* dataset, GACH turned out to be applicable in many application domains.

Table 3. Performance of GACH on *Wine*

	GACH	ITCH	RIC	OPTICS	SL
NMI	**0.7886**	0.7615	0.0000	0.5079	0.3852
AMI	**0.7813**	0.6912	0.0000	0.4817	0.3485
PREC	0.9401	**0.9737**	0.1591	0.7466	0.5309
REC	**0.9326**	0.8596	0.3989	0.6966	0.5337
DOM	0.3631	0.3285	**1.1405**	0.6853	1.0740

5 Conclusion

We proposed GACH – a genetic algorithm for finding cluster hierarchies, that combines the benefits of genetic algorithms, information theory and model-based clustering being an efficient and accurate hierarchical clustering technique. As GACH uses a MDL-based fitness function, it can be easily applied to real world applications without requiring any expertise about the data, like e.g. the real number of clusters. By the integration of an EM-like strategy, the content of all clusters is described by an intuitive model. GACH handles outliers by assigning them to appropriate inner nodes of the hierarchy, depending on their degree of outlierness. Our experimental evaluation demonstrates that GACH outperforms a multitude of other clustering approaches.

References

1. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering Points To Identify the Clustering Structure. In: SIGMOD Conference, pp. 49–60 (1999)
2. In: Bäck, T. (ed.) Proceedings of the 7th International Conference on Genetic Algorithms, East Lansing, MI, USA, July 19-23. Morgan Kaufmann, San Francisco (1997)

3. Böhm, C., Faloutsos, C., Pan, J.Y., Plant, C.: Robust Information-theoretic Clustering. In: KDD, pp. 65–75 (2006)
4. Böhm, C., Fiedler, F., Oswald, A., Plant, C., Wackersreuther, B., Wackersreuther, P.: ITCH: Information-Theoretic Cluster Hierarchies. In: ECML/PKDD, vol. (1), pp. 151–167 (2010)
5. Chardin, A., Pérez, P.: Unsupervised Image Classification with a Hierarchical EM Algorithm. In: ICCV, pp. 969–974 (1999)
6. Demiriz, A., Bennett, K.P., Embrechts, M.J.: Semi-supervised clustering using genetic algorithms. In: Artificial Neural Networks in Engineering, pp. 809–814 (1999)
7. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological) 39(1), 1–38 (1977)
8. Dom, B.: An Information-Theoretic External Cluster-Validity Measure. In: UAI, pp. 137–145 (2002)
9. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: KDD, pp. 226–231 (1996)
10. Filho, J.R., Alippi, C., Treleaven, P.: Genetic Algorithm Programming Environments. IEEE Computer 27, 28–43 (1994)
11. Goldberger, J., Roweis, S.T.: Hierarchical Clustering of a Mixture Model. In: NIPS (2004)
12. Grünwald, P.: A tutorial introduction to the minimum description length principle. CoRR math.ST/0406077 (2004)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explorations 11(1), 10–18 (2009)
14. Hamerly, G., Elkan, C.: Learning the k in k-means. In: NIPS (2003)
15. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge (1992)
16. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
17. Krishna, K., Murty, M.N.: Genetic K-means algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B 29(3), 433–439 (1999)
18. Lorena, L.A.N., Furtado, J.C.: Constructive Genetic Algorithm for Clustering Problems. Evolutionary Computation 9(3), 309–328 (2001)
19. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, Heidelberg (1996)
20. Nguyen, X.V., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: ICML, p. 135 (2009)
21. Pal, S.K., Bhandari, D., Kundu, M.K.: Genetic algorithms for optimal image enhancement. Pattern Recogn. Lett. 15(3), 261–271 (1994)
22. Pelleg, D., Moore, A.W.: X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In: ICML, pp. 727–734 (2000)
23. Pernkopf, F., Bouchaffra, D.: Genetic-Based EM Algorithm for Learning Gaussian Mixture Models. IEEE Trans. Pattern Anal. Mach. Intell. 27(8), 1344–1348 (2005)
24. Scheunders, P.: A genetic c-Means clustering algorithm applied to color image quantization. Pattern Recognition 30(6), 859–866 (1997)
25. Vasconcelos, N., Lippman, A.: Learning Mixture Hierarchies. In: NIPS, pp. 606–612 (1998)
26. Whitley, L.D., Starkweather, T., Bogart, C.: Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Computing 14(3), 347–361 (1990)