Niching an Estimation-of-Distribution Algorithm by Hierarchical Gaussian Mixture Learning

S.C. Maree
Academic Medical Center
Amsterdam, The Netherlands
s.c.maree@amc.uva.nl

T. Alderliesten
Academic Medical Center
Amsterdam, The Netherlands
t.alderliesten@amc.uva.nl

D. Thierens
Utrecht University
Utrecht, The Netherlands
d.thierens@uu.nl

P.A.N. Bosman
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
peter.bosman@cwi.nl

ABSTRACT

Estimation-of-Distribution Algorithms (EDAs) have been applied with quite some success when solving real-valued optimization problems, especially in the case of Black Box Optimization (BBO). Generally, the performance of an EDA depends on the match between its driving probability distribution and the landscape of the problem being solved. Because most well-known EDAs, including CMA-ES, NES, and AMaLGaM, use a uni-modal search distribution, they have a high risk of getting trapped in local optima when a problem is multi-modal with a (moderate) number of relatively comparable modes. This risk could potentially be mitigated using niching methods that define multiple regions of interest where separate search distributions govern sub-populations. However, a key question is how to determine a suitable number of niches, especially in BBO. In this paper, we present a novel, adaptive niching approach that determines the niches through hierarchical clustering based on the correlation between the probability densities and fitness values of solutions. We test the performance of a combination of this niching approach with AMaLGaM on both new and well-known niching benchmark problems and find that the new approach properly identifies multiple landscape modes, leading to much better performance on multi-modal problems than with a non-niched, uni-modal EDA.

CCS CONCEPTS

•**Mathematics of computing** → *Evolutionary algorithms; Continuous optimization;* •**Computing methodologies** → *Mixture modeling;*

KEYWORDS

continuous optimization, black box, estimation of distribution algorithm, Gaussian mixture model, hierarchical clustering

GECCO '17, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3071178.3071283

ACM Reference format:
S.C. Maree, T. Alderliesten, D. Thierens, and P.A.N. Bosman. 2017. Niching an Estimation-of-Distribution Algorithm by Hierarchical Gaussian Mixture Learning. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071283

1 INTRODUCTION

Sometimes, optimization problems require taking a Black Box Optimization (BBO) perspective, meaning that little to no information is assumed to be known about the problem at hand. In case of problems with real-valued variables, Estimation-of-Distribution Algorithms (EDAs) are applied with quite some success, but their performance heavily depends on the match between the search distribution and the problem landscape, which is unknown in BBO [5]. Finding out first what types of problem features exist so that a good matching EDA can be chosen is often cumbersome and time consuming. There is therefore a need for easy-to-use algorithms that perform well even when the problem landscape exhibits features such as multi-modality and correlation between problem variables.

EDAs for real-valued optimization problems are often based on a Gaussian distribution (in e.g., AMaLGaM [5], CMA-ES [10], and NES [19]). Besides many advantages, a limitation is that the Gaussian distribution is uni-modal, limiting the optimization to only one region of interest at a time. This is inefficient in case of a multi-modal landscape when the EDA tries to model multiple regions of interest, or niches, simultaneously, with a large risk of ending up in a local optimum, while a global optimum is desired. Generally, the probability of finding a global optimum can be increased by increasing the population size. In a highly multi-modal landscape however, this might not help, as a case study in [4] shows.

A logical next step is therefore to replace the Gaussian distribution by a mixture model, which overcomes both of these drawbacks. A mixture model is a weighted sum of probability distributions, referred to as mixture components, where each of these components governs a sub-population. It is however non-trivial to set the number of mixture components in advance, especially in BBO. Previous attempts of using mixture models in EDAs are [8], where the number of niches increases when solutions of high fitness are *far away* from the current niches, which is a threshold that needs to be set in advance, and [7], where the number of niches has to be set by the user.

An EDA based on a mixture model has many similarities to niching approaches in multi-modal optimization. These approaches are generally a wrapper around a core search algorithm that orchestrates the use of multiple instances of said algorithm in distinct locally interesting regions, or niches, of the search space. A recent overview and comparison of niching approaches is given in [1], where the Covariance Matrix Self-Adaptation Evolution Strategy with Repelling Sub-populations (RS-CMSA) is introduced, which is the winner of the GECCO'16 Competition on Niching Methods for Multimodal Optimization based on the CEC'2013 niching benchmark [12]. In RS-CMSA, but also in LIPS [17], NSDE [18], PNPCDE [3], NEA2 [16], and IPOP-CMA-ES [2], the user has to estimate the desired number of optima or a minimum distance between niches, and a considerable amount of effort is spent on calibration and sensitivity analysis of this parameter for different benchmark problems. Although insightful, such tuning does not necessarily provide a good basis for generalization to other problems.

In this work, we present an adaptive niching approach based on a Gaussian mixture model that determines the number of niches automatically based on correlation between probability densities and fitness values of solutions through hierarchical clustering.

The remainder of this paper is organized as follows. In Section 2, the Gaussian mixture model is introduced. In Section 3, we develop Hierarchical Gaussian Mixture Learning (HGML), which we apply use in an EDA in Section 4. In Section 5 we show experimentally that the probability of finding the global optimum increases when the population size increases, in contrast to a uni-modal EDA. Furthermore, we show that HGML can be applied when multiple global optima are desired. The results and further possible extensions are discussed in Section 6, and we conclude the paper in Section 7.

2 THE GAUSSIAN MIXTURE MODEL

Let $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ be the d-dimensional Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$, covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, and probability density function (pdf) in $\mathbf{x} \in \mathbb{R}^d$ given by,

$$f(\mathbf{x};\theta) = \frac{1}{|2\pi\Sigma|}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}, \tag{1}$$

where $\theta = \{\boldsymbol{\mu}, \Sigma\}$ are the parameters that uniquely define the Gaussian distribution. To fit a Gaussian distribution to a set of data points, one could use the Maximum Likelihood Estimator (MLE), which yields closed form solutions [9, 11] for the sample mean $\hat{\boldsymbol{\mu}}$ and sample covariance matrix $\hat{\Sigma}$,

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=0}^{N-1}\mathbf{x}_i, \quad \hat{\Sigma} = \frac{1}{N}\sum_{i=0}^{N-1}(\mathbf{x}_i-\hat{\boldsymbol{\mu}})(\mathbf{x}_i-\hat{\boldsymbol{\mu}})^\top. \tag{2}$$

A natural extension of the Gaussian distribution is the Gaussian Mixture Model (GMM),

$$f_K(\mathbf{x};\Theta) = \sum_{k=0}^{K-1} w_k f(\mathbf{x};\theta_k), \tag{3}$$

with K mixture components, where $\Theta = \{(w_k,\theta_k)\}_{k=0,\ldots,K-1}$ is the set of distribution parameters and w_k are the positive mixing weights, summing up to one. For a GMM, there is no closed form MLE, but estimates can for instance be found using the expectation-maximization algorithm [9]. This is however a computationally expensive algorithm, even for a fixed number of components K. It is computationally cheaper to first cluster the data into K clusters, and subsequently use the MLE of Eq. (2) to estimate a mixture component on each of the clusters. Due to this simplification, overlapping mixture components are no longer possible. For niching approaches, this is a desirable result. For breaking-up non-linear relations between problem variables, this is less desirable.

A fundamental difficulty of mixture models is the determination of the number of mixture components K, since more mixture components will inherently result in a better fit of the data [11]. How well the model fits the data can be quantified by information-theory based methods such as the Bayesian information criterion or two models can be compared using the likelihood ratio test. Both require a threshold to be set, and the best number of mixture components is found if adding another component does not improve its value more than a predefined threshold. There are two weaknesses associated with these approaches. First, the threshold needs to be calibrated, which is difficult in BBO, especially for different problem dimensionalities. Second, it might be that one or two mixture components do not fit the data well, but that three components fit very well. This will not be detected by these approaches.

In our case, rather than fitting the data , the primary target is to fit a model such that individuals with high fitness will be sampled with high probability. We will use this idea to determine a suitable number of mixture components.

3 HIERARCHICAL GAUSSIAN MIXTURE LEARNING

The starting point of the Hierarchical Gaussian Mixture Learning (HGML) algorithm is a set $\mathcal{S}$ of N solutions $\mathbf{x}_i \in \mathbb{R}^d$, where d is the problem dimensionality, i.e., the number of problem variables. Generally, $\mathcal{S}$ is the selection in an EDA, containing only high-fitness solutions, on which we learn the GMM. We apply a bottom-up clustering approach that is similar to the Unweighted Pair Group Method with Arithmetic Mean (UPGMA) approach [11]. Clustering starts with the cluster set $\mathcal{L}_0 = \{C_0,\ldots,C_{N-1}\}$ consisting of N clusters, each containing only a single solution. Then, iteratively the two closest clusters are merged, where merging two clusters is nothing more than grouping their solutions together. This is repeated until only one cluster is left, containing all solutions. After each merge, a new cluster set is saved. The result is a cluster tree $\mathcal{H} = \{\mathcal{L}_0,\ldots,\mathcal{L}_{N-1}\}$, where the j^{th} cluster set $\mathcal{L}_j$ contains all of $\mathcal{S}$, clustered in $N-j$ clusters.

The mean $\boldsymbol{\mu}_C$ of a cluster C is the mean of the solutions in C in the search space. The distance between two clusters C_0 and C_1 is defined as the Euclidean distance between their cluster means $\Delta(C_0,C_1) = \left\|\boldsymbol{\mu}_{C_0}-\boldsymbol{\mu}_{C_1}\right\|_2$. If C_0 has at least one solution with higher fitness than all solutions in C_1, we say that C_0 is better than C_1. Note that after merging two clusters C_0 and C_1 into C_2, the means can be updated using,

$$\boldsymbol{\mu}_{C_2} = \frac{|C_0|\boldsymbol{\mu}_{C_0}+|C_1|\boldsymbol{\mu}_{C_1}}{|C_0|+|C_1|}. \tag{4}$$

When merging clusters, the merge order is important. If there are two basins of attraction next to each other, clustering solely on the distance in the search space might wrongly determine that all

the solutions belong to the same basin of attraction. To prevent this, we use a distance measure based on the so-called nearest better tree [15].

A directed tree is constructed, where clusters are the nodes of the tree. From each cluster, an edge is formed to the nearest cluster that has higher fitness. This is repeated for each cluster. Since edges only go from better to worse clusters, cycles are not possible, resulting in a directed tree of clusters. The rationale is that a *long* edge between clusters suggests that two clusters are unlikely to be part of the same basin of attraction. Instead of considering all clusters for merging, we merge the two nearest neighbouring clusters in the nearest better tree, which both reduces computational effort and makes an attempt to prevent adjacent basins of attraction to be accidentally merged.

After merging two clusters, the nearest better tree is updated in a greedy way by replacing the two just merged clusters by the new one, similar to the merging procedure in [13]. Consequently, the length of all involved edges is recomputed using Eq. (4).

Previously, in related work [15, 16], the nearest better tree was defined as a tree connecting solutions, not clusters of solutions as is done in this work. The rationale behind the nearest better tree was then that a *long* edge between solutions suggests that these two solutions belong to the basin of attraction of two different local optima. A threshold is defined to determine which edges are too long, and when these long edges are removed, a number of clusters is obtained. From each of these, an instance of the core search algorithm could be run. It is however not straightforward to calibrate the threshold, as it needs to be adjusted for both the problem dimensionality and the population size. In this work, no threshold is defined in advance, but at each cluster level, first a GMM is fitted, and next the *most suitable* GMM is selected.

Pseudo code for hierarchical clustering is given in Algorithm 1.

Algorithm 1: Hierarchical Clustering

function : $\mathcal{H}$ = hierarchical_clustering($\mathcal{S}$)
input : Set $\mathcal{S}$ of N solutions
output : Hierarchical cluster tree $\mathcal{H} = \{\mathcal{L}_0, \ldots, \mathcal{L}_{N-1}\}$

Sort the solutions $\mathbf{x}_i \in \mathcal{S}$ on fitness value, fittest first;
Initialize a cluster C_i from each solution $\mathbf{x}_i$ in $\mathcal{S}$;
Initialize an empty set of edges $\mathcal{E}$;
for $i = 1, \ldots, N-1$ **do**
  $j^* = \arg\min_{j \in \{0,\ldots,i-1\}} \Delta(C_i, C_j)$;
  Create edge $e_{i-1} = (i, j^*, \Delta(C_i, C_j))$;
  Add e_{i-1} to $\mathcal{E}$;
end
Initialize the cluster set $\mathcal{L}_0 = \{C_0, \ldots, C_{N-1}\}$;
Add $\mathcal{L}_0$ to $\mathcal{H}$;
for $n = 0, \ldots, N-2$ **do**
  Find the shortest edge $e^- = (i^-, j^-, \Delta(C_{i^-}, C_{j^-}))$ in $\mathcal{E}$;
  Create a new cluster $C_{N+n} = C_{i^-} \cup C_{j^-}$;
  Create new cluster set $\mathcal{L}_{n+1} = \mathcal{L}_n \setminus \{C_{i^-}, C_{j^-}\} \cup C_{N+n}$;
  Add $\mathcal{L}_{n+1}$ to $\mathcal{H}$;
  Delete e^- from $\mathcal{E}$;
  Replace all occurrences of C_{i^-} and C_{j^-} in $\mathcal{E}$ by C_{N+n} using (4);
  Recompute edge lengths from and to C_{N+n};
end

3.1 Fitting the GMMs

After hierarchical clustering, we fit a GMM with parameter set Θ_j as in Eq. (3) to each cluster set $\mathcal{L}_j$ in the hierarchical cluster tree $\mathcal{H} = \{\mathcal{L}_0, \ldots, \mathcal{L}_{N-1}\}$. For a given cluster level j, the cluster set $\mathcal{L}_j$ consists of $N - j$ clusters. Fitting a GMM with $K = N - j$ mixture components as in Eq. (3) is cheap then, as we can simply use the MLEs from Eq. (2) to estimate the mean and covariance matrix $(\boldsymbol{\mu}_k, \Sigma_k)$ of each of the clusters. The mixing weights are chosen $w_k = \frac{1}{K}$, as we will sample from each mixture component equally. In order to apply the MLE for the covariance matrix in Eq. (2), at least $N_{\min} = 2$ solutions are required per cluster. If a full covariance matrix needs to be estimated, this is however not sufficient for a stable estimate, thus we force at least $N_{\min} = d + 1$ solutions in each cluster. We refer to a *valid cluster set* if all its clusters contain at least $N_{\min}$ solutions. The result is a set of GMMs, $\mathcal{G} = \{\Theta_{N-J^*}, \ldots, \Theta_{N-1}\}$ of J^* valid GMM models, all estimated on clusters of at least size $N_{\min}$, which implies $J^* \leq \lfloor N/N_{\min} \rfloor$. To select a single GMM out of the set $\mathcal{G}$, the density-fitness rank correlation measure is used.

3.2 Density-Fitness Rank Correlation

In [5], the Density-Fitness rank Correlation (DFC) was introduced as a tool to determine if the probability distribution of a real-valued EDA needs to be adjusted for the problem landscape. Here, we use that same idea to test which GMM is the best match to the structure of the problem at the current state of the search.

Given a distribution P, in our case a GMM with parameter set Θ as in Eq. (3) or a Gaussian distribution with parameter set θ as in Eq. (1), compute the probability density of each solution $\mathbf{x}_i \in \mathcal{S}$. Let $D_i(P)$ be the density rank under probability model P, such that the solution with the highest probability has rank 0. Let F_i be the fitness rank of that solution, such that the best solution has rank 0. Then, the DFC is given by the Spearman rank correlation $r_s(P)$ between the density- and fitness ranks,

$$r_s(P) = 1 - \frac{6 \sum_{i=1}^{N} [F_i - D_i(P)]^2}{N(N^2 - 1)}, \tag{5}$$

and takes values in $[-1, 1]$. The closer the DFC is to one, the higher the probability of sampling solutions with high fitness, thus the better the distribution fits the population. From the set of GMMs $\mathcal{G}$, we compute J^* DFCs, and the GMM with the highest DFC is chosen as the final one, but to improve efficiency and stability, a few adjustments are made. First of all, since $\mathcal{S}$ is clustered first and a mixture component is estimated on a cluster, we can compute the DFC per cluster. The total DFC of the GMM is then the average of the cluster DFCs, weighted by the cluster size. Second, GMM estimates are more prone to statistical noise in smaller clusters, thus if two clusters have a similar DFC, the larger cluster is preferable. To realize this, the DFCs are rounded with an accuracy of 0.05. If two rounded DFCs have the same value, the larger cluster is chosen. Third and final, if all the DFCs are negative, which implies that all of the estimated GMMs correlate poorly with the problem structure,

we fall back to using a single cluster that contains all solutions. Pseudo code of HGML is given in Algorithm 2.

Algorithm 2: Hierarchical Gaussian Mixture Learning

```
function : [Θ, L] = HGML(S)
input    : Set of solutions S
output   : GMM_Θ and corresponding cluster set L
H = hierarchical_clustering(S);
forall valid L_n ∈ H do
    DFC_n = 0;
    forall C_i ∈ L_n do
        Estimate θ_i = {μ_i, Σ_i} using (2) from C_i;
        DFC_n += r_s(θ_i)/|C_i| using (5);
    end
    Set Θ_n = {θ_i}_i;
end
n+ = arg max_n {Round(DFC_n, 0.05)};
if DFC_n+ < 0 then
    n+ = N − 1;
end
L = L_n+;
Θ = Θ_n+;
```

3.3 Computational Complexity of HGML

Let d be the problem dimensionality and N the population size on which the HGML is applied. Both the nearest better tree and the hierarchical cluster tree can be generated in $O(N^2 d)$. Since we have a lower bound of $N_{\min} = d + 1$ solutions per cluster, there are $O(Nd^{-1})$ valid cluster sets. Computing the MLE of the covariance matrix, which costs $O(Nd^2)$ per cluster, thus totals to $O(N^2 d)$ as well. Finally, in order to compute the probability density of solutions for the DFC, a Cholesky decomposition of the covariance matrix is required, which is an $O(d^3)$ operation and has to be performed for maximally $O(Nd^{-1})$ clusters, totalling to $O(Nd^2)$.

4 CLUSTERED AMALGAM

We use HGML described in Section 3 to niche the Adapted Maximum-Likelihood Gaussian Model Iterated Density-Estimation Evolutionary Algorithm (AMaLGaM-IDEA, or AMaLGaM for short) [5]. We refer to this niching algorithm as Clustered AMaLGaM, or CAMaLGaM. AMaLGaM was chosen as the core search algorithm partly because of its robust performance [6], but mainly because there are only few *algorithmic parameters* that need to be transferred over generations, making a first implementation relatively straightforward, allowing us to focus on the design and impact of using HGML.

We specifically use the EDA subtype of evolutionary algorithms, as a probability distribution is learned. We however note that our algorithm is more of a framework since how to generate new solutions from individual clusters can still be done in various ways. Pseudo code of the general outline of CAMaLGaM is shown in Algorithm 3, where a generation of AMaLGaM generates a new population and updates the algorithmic parameters as in the original AMaLGaM implementation.

Algorithm 3: General outline of CAMaLGaM

```
input  : Population size N
output : Set of high-fitness solutions S
O = uniform_sampling(N);
S = truncation_selection(O, τN);
while check_termination_criteria(S) do
    [Θ, L] = HGML(S);
    forall C_i ∈ L do
        [O_i, η_i] = AMaLGaM(C_i, η_i);
    end
    S = selection(O_0, ..., O_|L|−1);
end
```

4.1 Connecting Mixture Components over Generations

A crucial parameter in real-valued EDAs is the distribution multiplier, which prevents premature convergence due to limited diversity in the selection [6, 10]. We have to keep track of this and other algorithmic parameters over different generations and transfer them from mixture components in one generation to mixture components in the next generation.

In AMaLGaM, the set of algorithmic parameters η consist of three parameters: the distribution multiplier c; the no improvement stretch N_{nis}, which is the number of generations without improvement, and the previous mean $\boldsymbol{\mu}_{\text{old}}$ of the mixture component to compute the anticipated mean shift [6].

At the start of CAMaLGaM, the default algorithmic parameters are used, that is, $c = 1$, $N_{\text{nis}} = 0$, and $\boldsymbol{\mu}_{\text{old}}$ is set equal to the average of the initialization ranges of the search space. In subsequent generations, we merge the algorithmic parameters of the AMaLGaM instances along the hierarchical clustering. Recall that hierarchical clustering is initialized by clusters of size one, containing a single solution. We then also initialize the algorithmic parameters of these clusters to the algorithmic parameters of the cluster that the solution came from in the previous generation. When merging two clusters, the algorithmic parameters are updated as the weighted mean of the algorithmic parameters of the two parent clusters, similar to how the cluster mean is updated in Eq. (4).

4.2 Selection Schemes

We use two different selection schemes, depending on whether the user is interested in only global optima or a set of distinct local optima of high fitness.

4.2.1 Global selection. The first scheme is a global truncation selection scheme that selects the $\lfloor \tau N \rfloor$ best solutions *over all* clusters. In that way, local optima with a worse fitness are discarded during convergence. In this scheme, the algorithm will start with multiple clusters, and over time, clusters will disappear. If the user is interested only in global optima, this scheme should be used.

4.2.2 Local selection (niching). The second scheme is a local selection scheme that selects the $\lfloor \tau N_j \rfloor$ best solutions *in each* cluster C_j. Since clusters are non-overlapping, the resulting set of solutions will be a distinct set of locally optimal solutions. Note that even under the local selection scheme, due to the hierarchical clustering in each generation, the number of clusters will vary over time. If the user is interested in a set of distinct local optima with high fitness, this scheme can be used, as niches of high fitness are maintained over generations.

4.3 Termination Criteria

A run of CAMaLGaM is terminated when global termination criteria are met, like a maximum number of function evaluations, maximum runtime, or when the value-to-reach has been reached.

Moreover, if the variance of solutions reaches machine accuracy, either in search space or in fitness space, the DFC does not function and arbitrary clusterings occur. Therefore, clusters are terminated when this happens. The best solution of the terminated cluster is added to an archive and the cluster is removed from the population. If all clusters are terminated, CAMaLGaM is terminated as well. At the end of the run, the best solution of each cluster is also added to the archive and presented to the user.

5 EXPERIMENTS

Mixture models should be beneficial if the fitness landscape is multi-modal. In search of a benchmark problem that is adaptable both in the number of local optima and the problem dimensionality, we created a set of to-be-maximized multi-modal problems.

5.1 Sum of Gaussian Problems

The objective function of the Sum of Gaussian (SoG) problem with M local optima in d dimensions is defined as,

$$f_{\text{SoG}}^{d,M}(\mathbf{x}) = \sum_{m=0}^{M-1} f(\mathbf{x};\theta_m), \quad \theta_m = \{\boldsymbol{\mu}_m, h_m I\}, \tag{6}$$

where $f(\mathbf{x};\theta_m)$ is the Gaussian pdf in Eq. (1), h_m is a bandwidth parameter, and I is the identity matrix. The means $\boldsymbol{\mu}_m$ are sampled uniformly random on $[-1,1]^d$. The larger the bandwidth h_m, the larger the basin of attraction size of that peak, but also the lower the height of the peak.

The means are forced to be at least 0.1 away from the boundary of the sample domain, (thus effectively sampling the means on $[-0.9, 0.9]^d$) and at least a distance δ away from each other, using rejection sampling. The distance δ is chosen as large as possible, while still being able to sample a set of means using rejection sampling. If a suitable set of means is found, the bandwidth h_m is set a factor $\alpha \in (0, 0.5)$ times the distance between $\boldsymbol{\mu}_m$ and the nearest other peak, chosen such that the peak height is similar over different problem dimensionalities, which implies that α decreases when the dimensionality increases.

The SoG problem has a single global optimum and $M-1$ local optima, all of different height. The global optimum is very close to the $\boldsymbol{\mu}_m$ corresponding to the smallest bandwidth h_m, and can be found by initializing gradient descent or an EDA on the domain $\boldsymbol{\mu}_m \pm h_m$.

5.2 Success Rate vs Population Size

Increasing the population size of an EDA will generally increase the probability of avoiding local optima. However, in a multi-modal landscape with outspoken, approximately equal modes, such as in the SoG problems described above, this is often not sufficient, especially for single-population based EDAs. Other strategies are required, like restarts or initial clustering.

We benchmark AMaLGaM and CAMaLGaM on the SoG problems with dimensionalities $d = 2, 3, 5, 10, 20$ and $M = 2, 10, 40$ peaks. Fifty realizations of each SoG problem are generated, and optimized with AMaLGaM and CAMaLGaM for different population size, given by $N = 2^p(d+1)$, with population size factor $p = 1, 2, \ldots, 11$, within a given budget of $d \cdot 10^6$ function evaluations, or one hour runtime. As performance measure, the success rate is used, which is the fraction of runs in which the global optimum is successfully located with a difference less than 10^{-10} to the true optimum.

In Figure 1, the success rate of CAMaLGaM under both the global and local selection scheme is depicted, compared to the single-population-based AMaLGaM. The recommended population size for AMaLGaM, $N_{\text{AMaLGaM}} = \lfloor 17 + 3d^{1.5} \rfloor$ from [6], is indicated by the blue line in the right column, which is about sufficient for the two-peak problem, but is definitely too small for the higher multi-modal landscapes with 10 or 40 peaks.

The success rate increases when the population size increases, for both CAMaLGaM and AMaLGaM, however, for large problem dimensionality, the budget is spent, which can be observed for $d = 10, 20$ at $M = 10, 40$ peaks. Furthermore, the success rate decreases if the problem dimensionality or the number of peaks increases. We also observe that for problem dimensionalities 2, 3, and 5, CAMaLGaM reaches a success rate of 1.0, for both the global and local selection scheme, which is something we do not see for AMaLGaM. The success rate for AMaLGaM does increase for larger populations, but it stagnates, and does not reach 1.0 for any of the problems. Even though global optimization is performed, CAMaLGaM with the local selection scheme is slightly more effective. Especially for small populations, some clusters converge slower than others. With the global selection scheme, these slow converging clusters might get lost due to selection, which sometimes happens to be the cluster that was exploring the region around the global optimum. This effect is enhanced when the basin of attraction size differs between peaks. Local selection outperforms global selection because of this.

5.3 Adaptive Clustering

The following experiment is to show the added value of adaptive clustering, demonstrated on SoG problem, of which the peaks are within the domain $[-1,1]^d$. When initializing on a larger domain, say $[-10,10]^d$, there will initially only be a few samples in the domain of interest, which is not enough to determine the modality. In Figure 2, the average number of clusters over 50 runs is shown for $f_{\text{SoG}}^{5,10}$, with initialization on $[-10,10]^5$ and a population size of $N = 2^{10}(d+1)$. All three algorithms, CAMaLGaM with local and global selection, and AMaLGaM, are started with the same random seed, which is why the behaviour is identical in the first generations. After four generations, differences occur. AMaLGaM

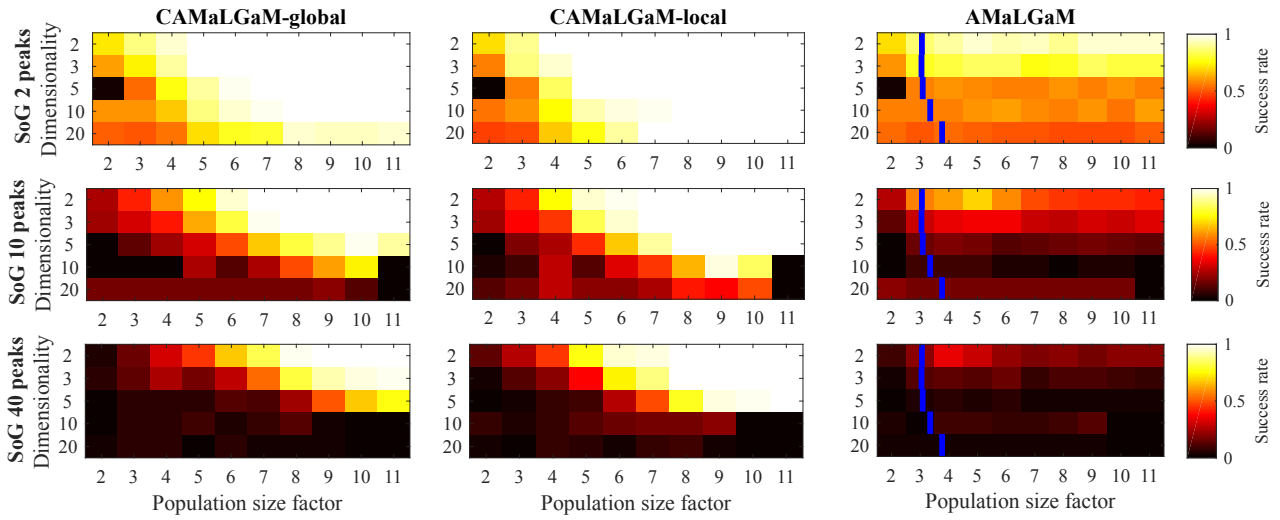

Figure 1: The fraction of trials successfully obtaining the global optimum out of 50 independent runs on the SoG problem, for dimensionalities $d = \{2, 3, 5, 10, 20\}$. The population size factor $p = \{2, \ldots, 11\}$ determines the population size by $N = 2^p(d + 1)$. The blue lines in the AMaLGaM plots (right column) represent the recommended population size of [6].

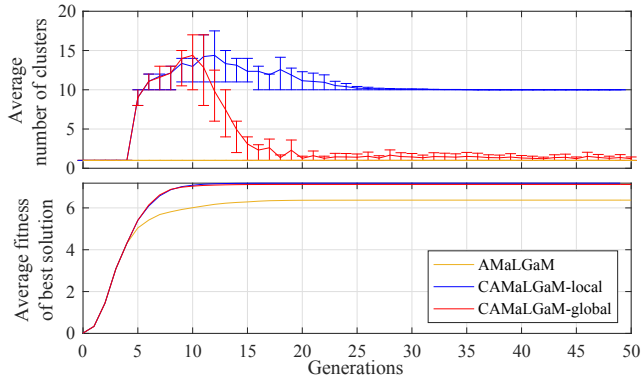

Figure 2: Performance of single-cluster (AMaLGaM) and adaptive clustering (CAMaLGaM) with both local and global selection on $f_{\text{SoG}}^{5,10}$ averaged over 50 independent runs. Error bars show the [0.25,0.75] quantile.

finds a local optimum more often, while CAMaLGaM always ends up in the global optimum. Local selection rapidly converges to 10 clusters, and keeps them, as desired, for the remainder of the run.

5.4 Global Optimization on Niching Benchmark

To verify the effectiveness of both the local and global selection schemes for finding global optima in multi-modal landscapes, we employ a selection of test problems of the CEC'2013 special session on multi-modal optimization [12] (see Table 1) that have a similar problem structure as the SoG problems. These are the Uneven Decreasing Maxima problem (PID 3), which is a $d = 1$ function with 4 local optima and a single global optimum, and the Shubert problem in $d = 2$ and $d = 3$ (PID 6 and 8). The Shubert problem has many global optima, but hundreds of local optima, without a global problem structure. We perform 50 runs of each algorithm, and record the fraction of runs successfully obtaining a global optimum. We repeat this with different population sizes, scaling again with a population size factor p such that $N = 2^p(d + 1)$. The maximum number of evaluations is limited, according to Table 1.

In Figure 3, the success rate of finding at least a single global optimum is reported. We observe that CAMaLGaM with global selection reaches a success rate of 1.0 for all problems at a population size factor $p = 9$, which corresponds to population sizes of 1024, 1536, and 2048 for respectively $d = \{1, 2, 3\}$.

Interesting is the behaviour of AMaLGaM, which performs best for $p = 3$, but the success rate decreases when the population size increases. This is in line with recent findings on solving a specific multi-modal optimization problem with uni-modal EDAs [4]. Similar behaviour is observed for CAMaLGaM with the local selection scheme on the Shubert 3D problem (PID=8), albeit to lesser extent, due to the mismatch between the local selection scheme and the global optimization objective. Because of the hundreds of local optima, when the population size increases, HGML recognizes these local peaks, and many evaluations are wasted in maintaining and exploring them. For a small population size, there are not sufficient solutions in each local peak, and the clustering falls back to using a single cluster to cover multiple peaks, which is actually the desired behaviour in this example. CAMaLGaM with the global selection scheme is the only algorithm that always succeeds for sufficiently large p. Since this behaviour was the primary goal of the design of HGML, we believe these result to be promising.

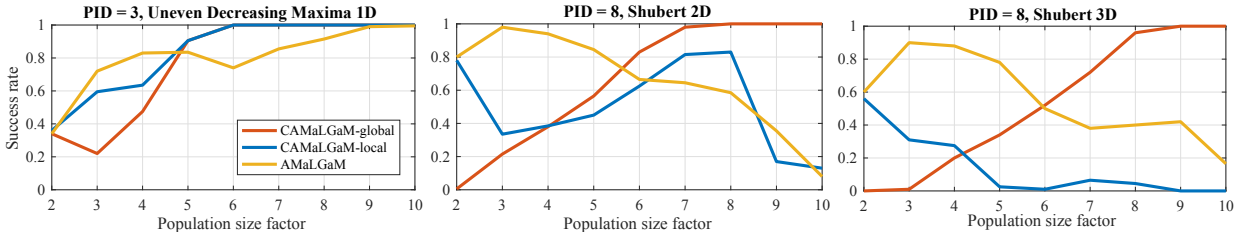

Figure 3: Rate of successfully obtaining at least one of the global optima for a selection of the benchmark problems from Table 1, with population size $N = 2^p(d+1)$ for increasing population size factor p.

5.5 Niching

In previous experiments, the performance of CAMaLGaM is shown on locating a single-best optimum. As a secondary goal, it is interesting to see how well HGML can aid in true multi-modal optimization, without further finetuning. We therefore show its performance in finding all global optima on the benchmark problems from the CEC'2013 special session on multi-modal optimization [12] in Table 1, and repeat the experiment as presented in [1]. As performance measure, the peak ratio is used, which is the number of global optima, referred to as peaks, correctly determined within the computational budget (MaxEvals). A peak is correctly detected if it is within an accuracy of ε_f of the true optimum, and we consider the accuracies $10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5}. The average peak ratio is then the peak ratio averaged over all accuracies. Each experiment is repeated 50 times.

CAMaLGaM was designed to be a one-run algorithm, but to make comparisons to competitors easier, we also test a version with independent restarts. CAMaLGaM is run until convergence, and as long as there is budget left, new, independent, runs are started.

Average peak ratios for the best performing population size factors, ranging from $p = 2$ to $p = 10$, are shown in Figure 4. For reference, the results of RS-CMSA [1], the winner of the GECCO'16 benchmark, and NEA2 [16], which is based on the nearest better tree, are shown.

For most of the problems, a single run performs as well as a restart strategy, which was the purpose of CAMaLGaM. For some problems, independent restarts are sufficient, but for others, an informed restart, like in RS-CMSA, where restarts are less likely to sample in previously explored areas, would improve performance. This is especially true for problems with different basin of attraction sizes.

The average peak ratio for low-dimensional problems is high, but decreases for larger problem dimensionalties, on which it is outperformed by the two reference algorithms. Still, considering it was not the main design purpose of CAMaLGaM to perform true multi-modal optimization, nor that it was specifically equipped with a tuned restart scheme, the results may again well be considered promising.

PID	Function	d	#gopt	MaxEvals
1	Five-Uneven-Peak Trap	1	2	50 000
2	Equal Maxima	1	5	50 000
3	Uneven Decreasing Maxima	1	1	50 000
4	Himmelblau	2	4	50 000
5	Six-Hump Camel Back	2	2	50 000
6	Shubert	2	18	200 000
7	Vincent	2	36	200 000
8	Shubert	3	81	400 000
9	Vincent	3	216	400 000
10	Modified Rastrigin	2	12	200 000
11	Composition Function 1	2	6	200 000
12	Composition Function 2	2	8	200 000
13	Composition Function 3	2	6	200 000
14	Composition Function 3	3	6	400 000
15	Composition Function 4	3	8	400 000
16	Composition Function 3	5	6	400 000
17	Composition Function 4	5	8	400 000
18	Composition Function 3	10	6	400 000
19	Composition Function 4	10	8	400 000
20	Composition Function 4	20	8	400 000

Table 1: Niching benchmark problems directly adopted from the CEC'2013 special session on multi-modal optimization [12]. For each Problem ID (PID) the function name, problem dimensionality d, number of global optima $\#gopt$, and function-evaluation budget MaxEvals are given.

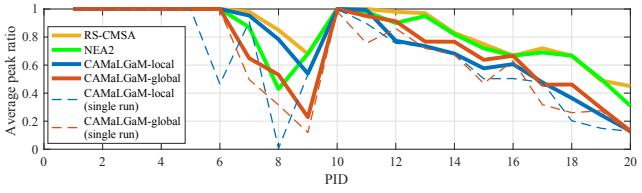

Figure 4: Average peak ratio on the benchmark problems from Table 1 for the best performing population size factor $p = p^*$ for each problem and algorithm.

6 DISCUSSION AND FUTURE WORK

In the previous section, we demonstrated on different benchmark problems that CAMaLGaM, equipped with HGML and global selection outperforms AMaLGaM in finding the global optimum on problems with many local optima. Furthermore, the probability of success increases when the population size is increased, something that was not observed for AMaLGaM. It is therefore of interest to equip CAMaLGaM with an interleaved multistart scheme with increasing population sizes like [14].

We introduced a parameter by rounding the DFC to reduce statistical noise, but a sensitivity analysis needs to be performed.

When testing the niching capacities of CAMaLGaM, a number of interesting aspects can be noted. First of all, the order in which clusters are considered for merging is of importance due to the fact that in HGML a heuristic clustering is applied based on the nearest better tree. Hence, other potential clusters merges are not considered. It could be of benefit to perform a clustering in which merges are considered that maximally increase the DFC, although this may also be computationally expensive.

Furthermore, when the goal is global optimization, the local selection scheme is inefficient as it explores local optima with low fitness. On the other hand, the global scheme might be too crude, and in multi-modal problem landscapes with varying basin-of-attraction size, peaks might get lost due to selection pressure. However, in contrast to AMaLGaM, increasing the population size could prevent this. Therefore, finding a good balance between the optimization objective and the selection scheme would improve the performance of CAMaLGaM, especially on problems with many local optima.

Similarly, when a problem has many local optima of low fitness, the DFC tries to model all of the local optima, while these are actually of too low fitness to be of relevance. The DFC measure needs to be adapted in order to overcome this.

Another issue is when improvements are found due to a generational shift. In CMA-ES this is the evolution path, in AMaLGaM, this is the anticipated mean shift. The DFC is ignorant for cases in which a normal distribution does not fit well, but where improvements are still found due to a shift. A good indicator for this situation is however not readily available.

Finally, other means of increasing performance may be by replacing the core search algorithm, AMaLGaM, which we chose here for its ease of use and the fact that only a low number of algorithmic parameters needs to be transferred over generations. Alternatively, CMA-ES could be used, which has a smaller recommended population size, but requires more care as it has many more algorithmic parameters that need to be transferred. In the CEC'2013 niching benchmark, many more restarts may well be possible when using CMA-ES, potentially increasing performance. CMA-ES is also used in the reference algorithms, RS-CMSA and NEA2.

7 CONCLUSIONS

In this paper, Hierarchical Gaussian Mixture Learning (HGML) is introduced as an adaptive method to learn a Gaussian mixture model meant for use in model-based evolutionary algorithms such as real-valued EDAs. Contrary to earlier attempts, HGML effectively circumvents the difficulty of setting the number of mixing components by hand by automatically adapting it to the problem landscape online, during optimization. We have applied HGML to AMaLGaM, which results in Clustered AMaLGaM (CAMaLGaM), a multi-modal EDA that shows promising results on different multi-modal problems of dimensionality up to $d = 20$, when performing global optimization as well as multi-modal optimization. Most importantly we show that in a multi-modal landscape, increasing the population size increases the probability that CAMaLGaM finds a global optimum, which is not achieved by the single-population based AMaLGaM, making HGML a promising avenue for future research and a solid basis for a principled, novel approach to the design of mixture-model based evolutionary algorithms.

8 ACKNOWLEDGEMENTS

This work is part of the research programme IPPSI-TA with project number 628.006.003, which is financed by the Netherlands Organisation for Scientific Research (NWO) and Elekta. We would like to thank Mike Preuss for providing source code of NEA2+.

REFERENCES

[1] A. Ahrari, K. Deb, and M. Preuss. 2016. Multimodal Optimization by Covariance Matrix Self-Adaptation Evolution Strategy with Repelling Subpopulations. *Evolutionary Computation* (in press) (2016).

[2] A. Auger and N. Hansen. 2005. A restart CMA evolution strategy with increasing population size. *The 2005 IEEE Congress on Evolutionary Computation* 2 (2005), 1769–1776.

[3] S. Biswas, S. Kundu, and S. Das. 2014. An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution. *IEEE Transactions on Cybernetics* 44, 10 (2014), 1726–1737.

[4] P.A.N. Bosman and M. Gallagher. 2017. The importance of implementation details and parameter settings in black-box optimization: a case study on Gaussian estimation-of-distribution algorithms and circles-in-a-square packing problems. *Soft Computing* (in press) (2017).

[5] P.A.N. Bosman and J. Grahl. 2008. Matching inductive search bias and problem structure in continuous estimation-of-distribution algorithms. *European Journal of Operational Research* 185, 3 (2008), 1246–1264.

[6] P.A.N. Bosman, J. Grahl, and D. Thierens. 2013. Benchmarking Parameter-free AMaLGaM on Functions With and Without Noise. *Evolutionary Computation* 21, 3 (2013), 445–469.

[7] P.A.N. Bosman and D. Thierens. 2001. Advancing Continuous IDEAs with Mixture Distributions and Factorization Selection Metrics. In *Pelikan, M. and Stastry, K. organizers, Proceedings of the Optimization By Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference - GECCO-2001.* 208–212.

[8] M. Gallagher, M. Frean, and T. Downs. 1999. Real-valued Evolutionary Optimization using a flexible probability density estimator. In *Proceedings of the 1st Annual Converence on Genetic and Evolutionary Computation-Volume 1.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 840–846.

[9] M.R. Gupta and Y. Chen. 2010. Theory and Use of the EM Algorithm. *Foundations and Trends in Signal Processing* 4, 3 (2010), 223–296.

[10] N. Hansen and A. Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *IEEE Computational Intelligence Magazine* 9, 2 (2001), 159–195.

[11] T. Hastie, R. Tibshirani, and J.H. Friedman. 2009. *The Elements of Statistical Learning: Data mining, Inference, and Prediction.* Springer, New York, NY, USA.

[12] X. Li, A. Engelbrecht, and M.G. Epitropakis. 2013. *Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization.* Technical Report. Evolutionary Computation and Machine Learning Group, RMIT University, Australia. 1–10 pages.

[13] G.J. McLachlan and S. Rathnayake. 2014. On the number of components in a Gaussian mixture model. *WIREs Data Mining Knowledge Discovery* 4, 5 (2014), 341–355.

[14] J.C. Pereira and F.G. Lobo. 2015. A Java Implementation of Parameter-less Evolutionary Algorithms. *arXiv preprint arXiv:1506.08694* (2015).

[15] M. Preuss. 2010. Niching the CMA-ES via nearest-better clustering. *Proceedings of the 12th annual Conference companion on Genetic and Evolutionary Computation* (2010), 1711–1718.

[16] M. Preuss. 2012. *Improved Topological Niching for Real-Valued Global Optimization.* Springer Berlin Heidelberg, Berlin, Heidelberg, 386–395.

[17] B.-Y. Qu, P.N. Suganthan, and S. Das. 2013. Novel multimodal problems and differential evolution with ensamble of restricted tournament selection. *IEEE Transactions on Evolutionary Computation* 17, 3 (2013), 387–402.

[18] B.-Y. Qu, P.N. Suganthan, and J.-J. Liang. 2012. Differential evolution with neighborhood mutation for multimodal optimization. *IEEE Transactions on Evolutionary Computation* 16, 5 (2012), 601–614.

[19] D. Wierstra, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. 2014. Natural Evolution Strategies. *Journal of Machine Learning Research* 15 (2014), 949–980.