Multi-Objective Mixture-based Iterated Density Estimation Evolutionary Algorithms

Dirk Thierens *dirk.thierens@cs.uu.nl*
Peter A.N. Bosman *peter.bosman@cs.uu.nl*
Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

We propose an algorithm for multi-objective optimization using a mixture-based iterated density estimation evolutionary algorithm (MIDEA). The MIDEA algorithm is a probabilistic model building evolutionary algorithm that constructs at each generation a mixture of factorized probability distributions. The use of a mixture distribution gives us a powerful, yet computationally tractable, representation of complicated dependencies. In addition it results in an elegant procedure to preserve the diversity in the population, which is necessary in order to be able to cover the Pareto front. The algorithm searches for the Pareto front by computing the Pareto dominance between all solutions. We test our approach in two problem domains. First we consider discrete multi-objective optimization problems and give two instantiations of MIDEA: one building a mixture of discrete univariate factorizations, the other a mixture of tree factorizations. Secondly, we look at continuous real valued multi-objective optimization problems and again consider two instantiations of MIDEA: a mixture of continuous univariate factorizations, and a mixture of conditional Gaussian factorizations as probabilistic model.

1 Introduction

In classical evolutionary computation search is driven by two interacting processes: selection focuses the search to more promising points in the search space while mutation and crossover try to generate new and better points from these selected solutions. Efficient exploration requires that some information of what makes the parents good solutions needs to be transfered to the offspring solutions. If there were no correlation between the fitness of the parents and the offspring the search process would essentially be an unbiased random walk. Whether or not information is passed between parents and offspring depends on the representation and accompanying exploration operators. For mutation this is usually accomplished by letting it take small randomized steps in the local neighbourhood of the parent solution. Crossover recombines parts of two parent solutions which results in a more globally oriented exploration step. This broader exploration requires a careful choice of genotype representation and crossover operator. A common practice in the design of evolutionary search algorithms is to develop a number of representations and operators by using prior domain knowledge, and picking the best after a considerable number of experimental runs.

An alternative to this labour intensive task is to try to learn the structure of the search landscape automatically, an approach often called linkage learning (see for instance [8]). In a similar effort to learn the structure of the problem representation a number of researchers have taken a more probabilistic view of the evolutionary search process ([1, 2, 7, 9, 11, 13, 15, 14, 17]). The general idea here is to build a probabilistic model of the current parent population and learn the structure of the problem representation by inducing the dependence structure of the problem variables. The exploration operators mutation and crossover are now replaced by generating new samples according to this probabilistic model (for a survey see [16]). In [2] we have given a general algorithmic framework for this paradigm called iterated density estimation evolutionary algorithm (IDEA). In this paper we will propose an algorithm for multi-objective optimization within the IDEA framework called MIDEA. The probabilistic model build is a mixture distribution that not only gives us a powerful and computationally tractable rep-

resentation to model the dependencies in the population, but also provides us with an elegant method to preserve the diversity in the population, which is needed in order to be able to cover the Pareto front.

2 Multi-objective optimization

Optimization is generally considered to be a search process for optimal or near optimal solutions in some search space where it is implicitly assumed that given any arbitrary solution one can always tell which solution is preferred. However such a single preference criterion does not always exist. In multi-objective optimization problems different objective functions have to be optimized simultaneously. A key characteristic of multi-objective optimization problems is the existence of whole sets of solutions that cannot be ordered in terms of preference when only considering the objective function values. To formalize this we define a number of relevant concepts. Suppose we have a problem with k objective functions $f_i(\boldsymbol{x}), i = 1 \ldots k$ which - without loss of generality - should all be minimized.

1. *Pareto dominance:* a solution $\boldsymbol{x}$ is said to dominate a solution $\boldsymbol{y}$ (or $\boldsymbol{x} \rhd \boldsymbol{y}$) iff $\forall i \in \{1, \ldots, k\} : f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}) \bigwedge \exists i \in \{1, \ldots, k\} : f_i(\boldsymbol{x}) < f_i(\boldsymbol{y})$.

2. *Pareto optimal:* a solution $\boldsymbol{x}$ is said to be Pareto optimal iff $\nexists \boldsymbol{y} : \boldsymbol{y} \rhd \boldsymbol{x}$.

3. *Pareto optimal set:* is the set $\mathcal{PS}$ of all Pareto optimal solutions: $\mathcal{PS} = \{\boldsymbol{x} \mid \nexists \boldsymbol{y} : \boldsymbol{y} \rhd \boldsymbol{x}\}$.

4. *Pareto front:* is the set $\mathcal{PF}$ of objective function values of all Pareto optimal solutions: $\mathcal{PF} = \{\mathrm{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) \mid \boldsymbol{x} \in \mathcal{PS}\}$.

Note that the Pareto optimal set is defined in the parameter space, while the Pareto front is defined in the objective space. Multi-objective problems have been tackled with different solution strategies. A strategy which is particularly interesting from an evolutionary computation viewpoint is to search for the Pareto front - or for a representative set of Pareto optimal solutions - by making use of the Pareto dominance concept. The idea is to maintain a population of solutions that *cover* the entire Pareto front. The notion of searching a search space through maintaining a population of solutions is a key characteristic of evolutionary algorithms, which makes them natural candidates for multi-objective optimization algorithms following the *covering* strategy. The field of evolutionary multi-objective optimization has indeed seen an explosive growth in recent years (for a survey see [4]).

3 Multi-objective mixture-based IDEA

The IDEA is a framework for *Iterated Density Estimation Evolutionary Algorithms* that uses probabilistic models to guide the evolutionary search [2]. A key characteristic of this class of evolutionary algorithms is the way they explore the search space. Contrary to classical evolutionary algorithms who generate new offspring by applying crossover and mutation to individual parent solutions, IDEAs generate new offspring by sampling from a probability distribution $\hat{P}_{\varsigma}^{\theta_t}(\boldsymbol{\mathcal{Y}})$. The probability distribution $\hat{P}_{\varsigma}^{\theta_t}(\boldsymbol{\mathcal{Y}})$ is induced every generation from the $\lfloor \tau n \rfloor$ best performing individuals (n = population size, $0 < \tau < 1$). One way of achieving this, is by finding a factorized probability distribution. A factorized probability distribution is product of probability density functions (pdfs). Factorizations are usually composed either of multivariate joint pdfs or of multivariate conditional pdfs in which a single variable is conditioned on a multiple of others. The model of the probability distribution $\hat{P}_{\varsigma}^{\theta_t}(\boldsymbol{\mathcal{Y}})$ is determined in two steps. Firstly, a structure ς implying a factorization of the probability distribution needs to be determined. Secondly, a vector of parameters $\boldsymbol{\theta}$ need to be fitted. The choice of structure ς defines how the algorithm will explore the search space and whether it will be able to find good solutions efficiently. In this paper we will discuss four structures in the context of multi-objective optimization problems: a mixture of discrete univariate factorizations, a mixture of tree factorizations, a mixture of continuous univariate factorizations, and a mixture of conditional Gaussian factorizations. Assuming without loss of generality, we want to *minimize* $C(\boldsymbol{y})$. For every problem variable y_i we introduce a corresponding random variable Y_i and define $P^{\theta}(\boldsymbol{\mathcal{Y}})$ to be a probability distribution that is uniform over all vectors $\boldsymbol{\mathcal{Y}}$ with $C(\boldsymbol{y}) \leq \theta$. Sampling from $P^{\theta}(\boldsymbol{\mathcal{Y}})$ gives more samples that evaluate to a value below θ. To use this in an iterated algorithm, we select the best $\lfloor \tau n \rfloor$ samples in each iteration t and let θ_t be the worst selected sample cost. We then estimate the distribution of the selected samples and thereby find $\hat{P}_{\varsigma}^{\theta_t}(\boldsymbol{\mathcal{Y}})$ as an approximation to the true distribution $P^{\theta_t}(\boldsymbol{\mathcal{Y}})$. New samples can then be drawn from $\hat{P}_{\varsigma}^{\theta_t}(\boldsymbol{\mathcal{Y}})$ and be used to replace the worst $n - \lfloor \tau n \rfloor$ samples. This results in a *elitist* mechanism since we have a monotonically decreasing series $\theta_0 \geq \theta_1 \geq \ldots \geq \theta_{t_{\mathrm{end}}}$.

3.1 Factorization mixtures by clustering

The structure of the sample vector may be highly non-linear. This non-linearity can force us to use probabilistic models of a high complexity to retain some of

this non-linearity. However, especially using relatively simple probability density functions, the non-linear interactions cannot always be captured even with higher order models. The key issue is the use of *clusters*. The use of clusters allows us to efficiently break up non-linear interactions so that we can use simple models to get an adequate representation of the sample vector. Each cluster is processed separately in order to have a probability distribution fit over it. The resulting probability distribution is a weighted sum of the individual factorizations over each cluster:

$$\hat{P}_{\mathfrak{f}\langle\boldsymbol{\kappa}\rangle}(\boldsymbol{\mathcal{Y}}) = \sum_{i=0}^{|\mathfrak{K}|-1} \beta_i \hat{P}^i_{\mathfrak{f}_i}(\boldsymbol{\mathcal{Y}}) \tag{1}$$

An effective way to set the mixture coefficients β_i, is to proportionally assign larger coefficients to the clusters with a better average cost. Pelikan and Goldberg [14] proposed this method to introduce niching in the probabilistic model-building genetic algorithms. By taking the absolute value of the difference of the average cluster cost and the average initial sample vector cost, we allow for both maximization as well as minimization problems. Here we will make use of the randomized Euclidean leader algorithm which is one of the fastest clustering algorithms. The first sample to make a new cluster is appointed to be its leader. The leader algorithm goes over the sample vector exactly once. For each sample, it finds the first cluster that has a leader being closer to the sample than a given threshold $\mathfrak{T}_d$ (using the normalized Euclidean distance measure). If no such cluster can be found, a new cluster is created containing only this sample. To prevent the first clusters from becoming a lot larger than the later ones, we randomize the order in which clusters are inspected.

3.2 Multi-objective mixture-based IDEA

The algorithm discussed so far is still a single-objective optimization algorithm. To change it into a multi-objective Pareto covering optimization algorithm we need to make the following modifications:

1. First, we have to *search for the Pareto-front*: in the IDEA framework selection picks out the best $\lfloor \tau n \rfloor$ samples. Making this selection on the basis of Pareto dominance allows us to search for the Pareto front. For each individual in the population we determine the number of individuals by which it is dominated, calling this its *domination count*. All individuals are sorting according to increasing domination count and the top $\lfloor \tau n \rfloor$ solutions are selected.
2. Second, we have to *cover the Pareto-front*: maintaining diversity is needed to prevent the population to converge to a single Pareto optimal point instead of to a representative set of the entire front. Since the mixture-based IDEA already constructs a set of clusters we can simply use this to maintain the diversity. Note that the clustering can be done in the parameter space or in the objective space, but to maintain a good covering of the Pareto front clustering in the objective space is more suitable.

Finally, the multi-objective mixture-based iterated density estimation evolutionary algorithm - or MIDEA - can be summarized as:

MIDEA(n, τ)
1 Evaluate n randomly generated samples $\boldsymbol{\mathcal{P}}$
2 Iterate until termination
  2.1 Compute the domination counts
  2.2 Select the $\lfloor \tau n \rfloor$ best samples from $\boldsymbol{\mathcal{P}} \Rightarrow \boldsymbol{\mathcal{P}}^{\int}$
  2.3 Set θ_t to the worst selected cost
  2.4 Search $\boldsymbol{\mathcal{P}}^{\int}$ for a structure ς
  2.5 Estimate parameters $\boldsymbol{\theta} \overset{fit}{\Leftarrow} \varsigma \Rightarrow \hat{P}_{\varsigma}(\boldsymbol{\mathcal{Y}})$
  2.6 Draw $n - \lfloor \tau n \rfloor$ new samples from $\hat{P}_{\varsigma}(\boldsymbol{\mathcal{Y}})$
  2.7 Evaluate the new samples
  2.8 Add the new samples to $\boldsymbol{\mathcal{P}}^{\int} \Rightarrow$ new $\boldsymbol{\mathcal{P}}$

Depending on the type and complexity of the application one has to choose the kind of factorization learned during the structure search. To illustrate this we will implement four versions of the MIDEA algorithm: two for discrete and two for continuous multi-objective optimization problems.

3.2.1 MIDEA_univariate and MIDEA_tree

A simple structure one can apply for discrete problems is the mixture of univariate factorizations leading to the MIDEA_univariate algorithm. The probability a problem variable has a certain value is assumed to be independent of other problem variables. Although this is a very strong assumption it appears in practice that many problems can be solved this way. When optimizing more complicated problems it is necessary to learn more structure of the domain representation. One approach which is still simple enough to be computationally efficient is to model the domain variable interactions with a tree factorization. The model thus becomes a mixture of trees, a probability model recently proposed in [12]. The MIDEA_tree can be viewed as a generalization of the optimal dependency tree algorithm [1] towards a mixture model and adapted for multi-objective problems. Interestingly, the use of a mixture of tree factorizations in proba-

bilistic model building EAs is currently also proposed in relation with the Estimation Maximization learning algorithm [17].

3.2.2 MIDEA_univariate and MIDEA_Gaussian

For multi-objective continuous function optimization we can again use a mixture model where each component distribution ignores conditional dependences between the variables. Inducing a mixture of univariate factorizations is very simple and extremely fast. A more intelligent search can be performed when using a model that learns conditional dependences between the variables. Here we induce a mixture of conditionally factorized Gaussian probability density functions. This structure has the advantage of being capable to learn conditional dependences between variables, while at the same time being computationally efficient enough to be applied at each generation. Learning a conditional factorization from the vector of selected samples can be done in a variety of ways [16]. Here we use an incremental algorithm that starts from the empty graph with no arcs. Each iteration, the arc to add is selected as the arc that increases some metric the most. If no addition of any arc further increases the metric, the final factorization graph has been found. The metric used is the Bayesian Information Criterion (for details see [3]).
Without detailed knowledge about the functions it is not possible to tell which structure is optimal. To illustrate the potential of each model we ran a number of experiments on problems previously studied in the literature.

4 Experimental results

4.1 Multi-objective 0/1 knapsack problem

Our first test function is a discrete multi-objective 0/1 knapsack problem taken from Zitzler and Thiele [18] who introduced it to compare a number of different multi-objective evolutionary algorithms. The problem is additionally interesting because of its real life practicality and the large string lengths it requires. Whereas the problem definition is relatively simple, optimizing it is difficult ($\mathcal{NP}$-hard). The multi-objective 0/1 knapsack problem consists of a set of n_I items and a set of n_K knapsacks. With each knapsack i, a weight $w_{i,j}$ and a profit $p_{i,j}$ are associated with each item j. Each knapsack i has an upper-bound c_i on the amount of weight it can hold, which is called the *capacity constraint*. The objective is to fill each knapsack so that the profit of the selected items is maximized, but without violating the capacity constraints. If item i is selected, it is automatically placed in *every* knapsack. This creates a multi-objective interaction between the knapsacks: the goal is to search for a vector of decision variables $\boldsymbol{x} \in \{0,1\}^{n_I}$ such that each objective function f_i in $(f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{n_K-1}(\boldsymbol{x}))$ is maximised with

$$\forall_{i\in\{0,1,\ldots,n_K-1\}} \left[f_i(\boldsymbol{x}) = \sum_{j=0}^{n_I-1} p_{i,j}x_j \right]$$

$$\text{s.t. } \forall_{i\in\{0,1,\ldots,n_I-1\}} \left[\sum_{j=0}^{n_K-1} w_{i,j}x_j \leq c_i \right].$$

To deal with the feasibility problem with respect to the capacity constraints, we use a repair method. The type of repair method used has a great influence on the way the search space is traversed and thus on the performance of the optimization algorithm. To compare different algorithms with respect to their multi-objective performance, it is important to use the same repair method. To this end, we have used the same approach as is in [18]. If a solution violates a constraint, the repair algorithm iteratively removes items until all constrains are satisfied. The order in which the items are investigated, is determined by the maximum profit/weight ratio. The items with the lowest profit/weight ratio are removed first. This amounts to computing the quotients

$$q_j = \max_{i\in\{0,1,\ldots,n_K-1\}} \left\{ \frac{p_{i,j}}{w_{i,j}} \right\}$$

on beforehand and sorting the q_j.
The profits, weights and knapsack capacities are chosen as follows: $p_{i,j}$ and $w_{i,j}$ are random integers chosen from the interval [10,100], while the capacities c_i are set to half the items' weight in the corresponding knapsack:

$$c_i = 0.5 \sum_{j=0}^{N_I-1} w_{i,j}.$$

This results in half of the items to be expected in the optimal solutions. We performed tests on problems with two knapsacks ($n_K = 2$) allowing us to plot the Pareto front found by MIDEA and to make a visual comparison with results obtained by the Strength Pareto Evolutionary Algorithm (SPEA) and the Nondominated Sorting Genetic Algorithm (NSGA). In [18] a total of 8 algorithms were compared but for clarity we restrict ourselves here to SPEA and NSGA: they are the most commonly known and SPEA gave the best results of all 8 algorithms. Three different knapsack problems with an increasing number of items were

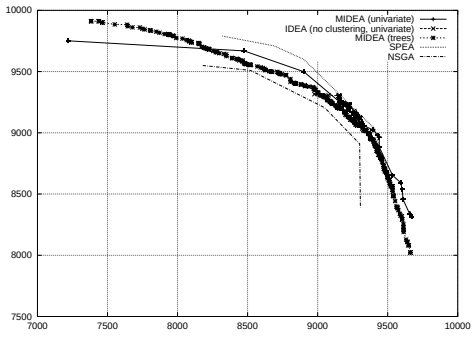

Figure 1: knapsack 250 items

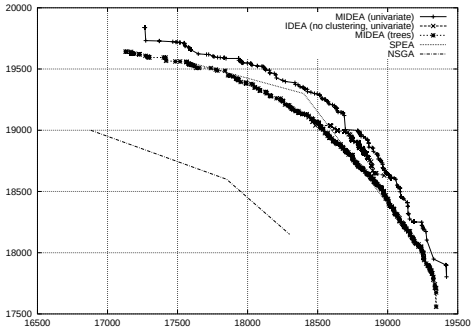

Figure 2: knapsack 500 items

studied, $n_I \in \{250, 500, 750\}$. The data sets are the same as those used by Zitzler and Thiele (available on http://www.tik.ee.ethz.ch/ zitzler/testdata.html).

In our experiments we fixed the selection size to $\lfloor \tau n \rfloor = 200$ ($\tau = 0.3$, population size $n = 667$). We have also fixed the number of evaluations to be the same as in the tests by Zitzler and Thiele, respectively $\{60000, 80000, 100000\}$ for increasing values of n_I. For the univariate factorization the final front reported is obtained by combining the results of 30 independent runs (similar to Zitzler and Thiele), but it should be noted that individual runs give an almost as wide covering of the Pareto front. Results for the tree factorization are only from one single run. The clustering is done in the objective space using the leader algorithm with $\mathfrak{T}_d$ chosen so as to get at least 5 clusters.

Figures 1, 2, and 3 show the results for the MIDEA with a mixture of univariate distributions, and for the MIDEA with a mixture of trees. To study the influence of the clustering we have also tested the univariate distributions without clustering. The graphs give also a rough sketch of the front found by SPEA and NSGA in [18]. A number of observations can be made:

1. The MIDEA algorithm found solutions with high objective function values. Performance is comparable to SPEA for the problem sets with 250 and 500 items. For the knapsack problem with 750 items MIDEA finds a Pareto front that clearly dominates the solutions found by SPEA. The results from NSGA are substantially worse.

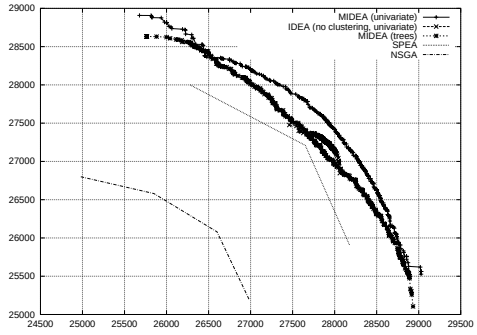

Figure 3: knapsack 750 items

2. The MIDEA algorithm found a widely covered Pareto front, wider than SPEA and NSGA.

3. Clustering is necessary for the covering to take place: without it the algorithm finds only a small part of the Pareto front.

4. On the two larger problem sets ($n_I \in \{500, 750\}$) the Pareto front obtained by the mixture of univariate distributions is slightly better than the front found by the mixture of trees. It is reasonable to assume that this is only an indication of the faster convergence speed of the mixture of univariate distributions. When more function evaluations would be done one might expect the mixture of trees algorithm to catch up, and even surpassing it for more difficult problems. This should be further investigated though.

4.2 Multi-objective continuous function optimization

Next to the multi-objective 0/1 knapsack problem we also tested the MIDEA algorithm on multi-objective continuous function optimization problems taken from the literature [5].

First we will look at the mixture of Gaussian pdfs using learning conditional factorizations. We fixed the selection size to $\lfloor \tau n \rfloor = 250$ ($\tau = 0.3$, population size $n = 834$). Clustering is done using the leader algorithm in both the objective space as well as the parameter space, with $\mathfrak{T}_d$ chosen so as to get approximately 3 clusters on the Pareto front. The final front reported is obtained by combining the results of 10

	MOP_2	MOP_4	EC_4	EC_6
Obj.	3754	10762	1019330	9535
Par.	3754	35348	2500000	8835
None	3754	43058	2500000	8426

Figure 4: Average number of evaluations.

independent runs. As before it should be noted that individual runs give an almost as wide covering of the Pareto front. The average number of required evaluations for each type of clustering is stated in figure 4. For comparison, we also tested an approach using *no* clustering. Termination is enforced when the domination count of all of the selected samples equals 0. At such a point, the selected sample vector contains only non-dominated solutions. Note that this does not have to imply at all that full convergence has been obtained since the front itself may not be optimal. To prevent the alternative of allowing an arbitrary number of generations or evaluations, a good termination criterion might be when non of the selected samples is dominated by any of the selected samples in the previous generation. For now, we restrict ourselves to the simple termination criterion, keeping in mind that premature convergence is possible. No single run was allowed more than $2\frac{1}{2} \cdot 10^6$ evaluations. Finally the conditional Gaussian factorizations are searched using the BIC metric with $\lambda = \frac{1}{2}$ (which makes this measure similar to the minimum description length measure).

Name	**Objectives**	**Domain**
MOP_2	$f_0 = 1 - e^{-\sum_{i=0}^{l-1}\left(y_i - \frac{1}{\sqrt{l}}\right)^2}$ $f_1 = 1 - e^{-\sum_{i=0}^{l-1}\left(y_i + \frac{1}{\sqrt{l}}\right)^2}$	$[-4, 4]^3$
MOP_4	$f_0 = \sum_{i=0}^{l-2} -10e^{-0.2\sqrt{y_i^2 + y_{i+1}^2}}$ $f_1 = \sum_{i=0}^{l-1} \lvert y_i \rvert^{0.8} + 5\sin(y_i^3)$	$[-5, 5]^3$
EC_4	$f_0 = y_0$ $f_1 = \gamma\left(1 - \sqrt{\frac{y_0}{\gamma}}\right)$ $\gamma = 91 + \sum_{i=1}^{l-1}\left(y_i^2 - 10\cos(4\pi y_i)\right)$	$[-1, 1] \times$ $[-5, 5]^9$
EC_6	$f_0 = 1 - e^{-4y_0}\sin^6(6\pi y_i)$ $f_1 = \gamma\left(1 - (\frac{f_0}{\gamma})\right)$ $\gamma = 1 + 9\left(\sum_{i=1}^{l-1} \frac{y_i}{9}\right)^{0.25}$	$[0, 1]^{10}$

In figures 5 and 6, the results using objective clustering on MOP_2 and MOP_4 are shown respectively. For each of these two problems, none of the individual runs differ significantly from the combined result. Moreover, the results of parameter clustering as well as no clustering at all are also similar to these results, so we omit further graphs for these two problems. The table in figure 4 indicates that using the MIDEA algorithm requires only a few evaluations to adequately solve the two MOP problems.

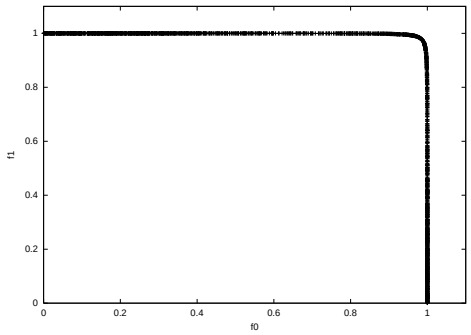

Figure 5: Result for MOP_2 (objective clustering).

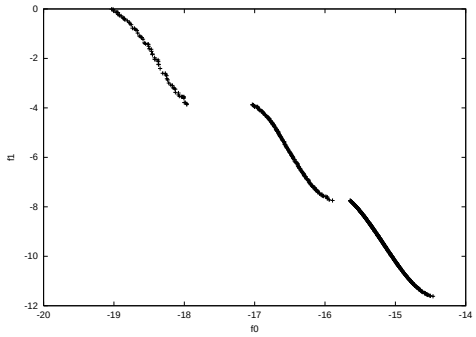

Figure 6: Result for MOP_4 (objective clustering).

Compared to EC_4, the two MOP problems are relatively simple. Converging to the optimal front is very difficult in EC_4. In figure 4, we can see that we indeed require a vastly larger number of evaluations. Only when we cluster in the objective space, do we on average require less than the maximum of $2\frac{1}{2} \cdot 10^6$ evaluations. However, closely observing the results points out that premature convergence has often taken place in the algorithm with objective clustering. Figure 7 shows the individual plots of each run. Taking more clusters in combination with a larger population size to effectively fill up and use these additional clusters, might lead to a better estimation of the promising regions of the multi-objective space. To illustrate this, we have plotted the resulting fronts after 10 runs for objective clustering with $\lfloor \tau n \rfloor = 500$ and $\mathfrak{T}_d$ such that we have approximately 5 clusters, and for parameter clustering with $\lfloor \tau n \rfloor = 125$ and $\mathfrak{T}_d$ such that we have 7 clusters. The results in figure 8 show that very good results are obtained with these settings. It should also be noted that some sort of clustering is *crucial* to be able to tackle difficult problems such as EC_4: when no clustering is applied the results are rather poor even for large populations sizes.

The main difficulty with problem EC_6 is that the optimal front is not uniformly distributed in its solutions.

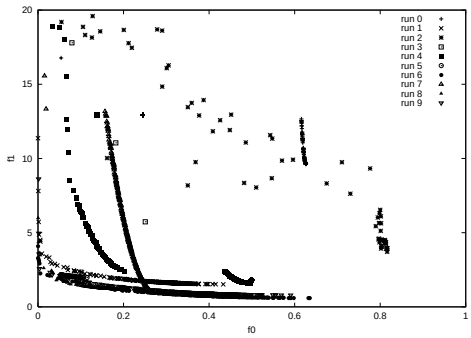

Figure 7: All runs for EC_4 (objective clustering).

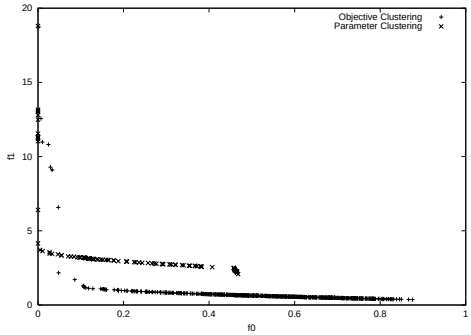

Figure 8: Results for EC_4.

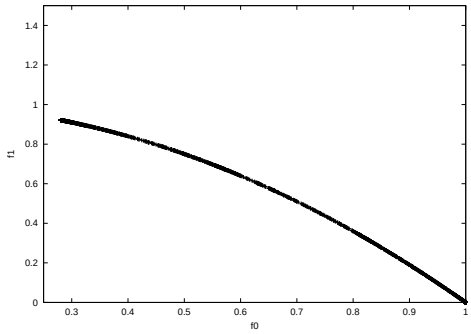

Figure 9: Result for EC_6 (objective clustering).

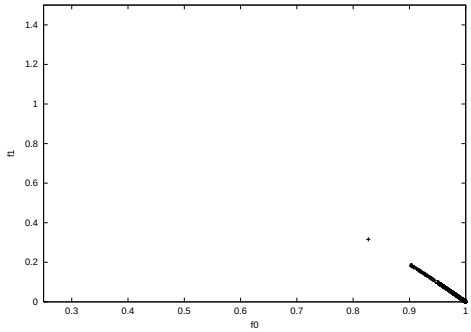

Figure 10: Result for EC_6 (parameter clustering).

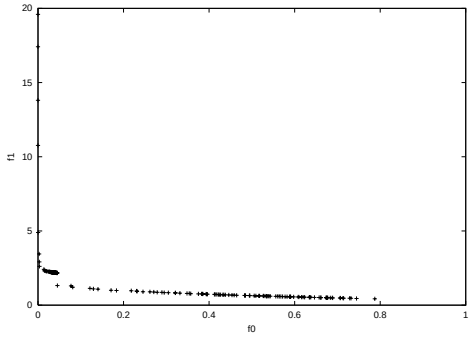

Figure 11: Result for EC_4 with univariate factorization (objective clustering).

Without clustering, we are therefore very likely to find only a part of the front. Furthermore, by clustering in parameter space, we also have no guarantee to find a good representation of the front since the parameter space is directly related to the density of the points along the front. On the other hand, clustering this space *does* give a means of capturing more regions than a single cluster can. If we are to cluster in the objective space, we should have no problem finding a larger part of the front unless the problem itself is very difficult as is the case for instance with EC_4. In figures 9 and 10, the results for objective clustering and parameter clustering are shown respectively. Using parameter clustering is clearly not effective. It should also be noted that the Pareto front found in figure 9 seems to coincide with the optimal Pareto front, which is not trivial to achieve since the fast elitist non-dominated sorting GA (NSGA-II [6]), the strength Pareto Evolutionary Algorithm (SPEA [18]), and the Pareto-archived evolution strategy (PAES [10]) are all reported to converge to a sub-optimal front ([6]).

In the experiments so far, the structure learned at each generation is a conditionally factorized Gaussian probability density function. It might well be possible that the fitness function can be optimized without the need to learn the conditional dependences between variables. In this case it would be computationally more efficient to use a probability density structure that ignores the interactions between the variables. To get a feeling of the impact of this choice we have optimized the functions EC_4 and EC_6 with a mixture of univariate factorizations. A population size of $\lfloor \tau n \rfloor = 125$, resp. $\lfloor \tau n \rfloor = 50$ ($\tau = 0.3$) was used, with a cluster threshold $= 1.5$, resulting in 3 to 5 clusters. Clustering was done in the objective space, and a total of 10 runs were performed. Figures 11 and 12 show that the Pareto front found is of similar quality than in the previous experiment using conditionally factorized Gaussian pdfs with objective clustering. The average

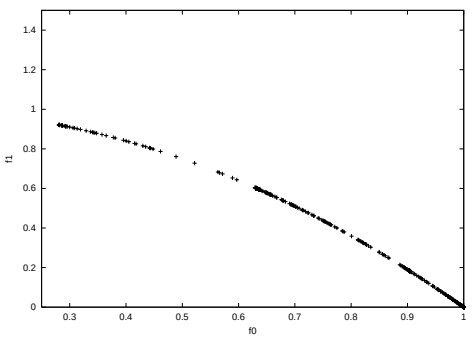

Figure 12: Result for EC_6 with univariate factorization (objective clustering).

amount of function evaluations for EC_4 was 209635, while for EC_6 the number was 2284. These figures are substantially lower than those found before (see figure 4), indicating that for these functions the computational effort spent by learning a more powerful and complicated model seems to be unnecessary. It should be noted that this gives only a rough impression about convergence speed and quality of the algorithms. Future studies will have to look at the influence of population size, selection threshold, and cluster size.

5 Conclusion

We have proposed the multi-objective mixture-based iterated density estimation evolutionary algorithm MIDEA. MIDEA builds a mixture distribution as probabilistic model resulting in a computational efficient method to represent complicated dependencies, and at the same time in an elegant procedure to search for a good covering of the Pareto front. As specific instantiations of the proposed algorithm we have implemented a mixture of univariate factorizations and a mixture of tree factorizations for discrete multi-objective optimization, and a mixture of continuous univariate factorizations and a mixture of conditional Gaussian factorizations for continuous optimization problems. Experiments showed good results for all models, including the simple univariate models.

References

[1] S. Baluja and S. Davies. Using optimal dependency–trees for combinatorial optimization: Learning the structure of the search space. In D.H. Fisher, ed., *Proc. of the 1997 Int. Conf. on Machine Learning*. Morgan Kauffman Pub., 1997.

[2] P.A.N. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, eds., *Parallel Problem Solving from Nature – PPSN VI*, pages 767–776. Springer, 2000.

[3] P.A.N. Bosman and D. Thierens. Mixed IDEAs. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2000/2000-45.ps.gz, 2000.

[4] C.A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.

[5] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[6] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer et al., eds., *Parallel Problem Solving from Nature*, pages 849–858. Springer, 2000.

[7] M. Gallagher, M. Fream, and T. Downs. Real–valued evolutionary optimization using a flexible probability density estimator. In W. Banzhaf et al., eds., *Proc. of the 1999 Genetic and Evolutionary Computation Conference*, p. 840–846. Morgan Kaufmann, 1999.

[8] D.E. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithm. In S. Forrest, ed., *Proc. of the 5th International Conference on Genetic Algorithms*, pages 56–64. Morgan Kaufmann, 1993.

[9] G. Harik, F. Lobo, and D.E. Goldberg. The compact genetic algorithm. In *Proc. of the 1998 IEEE Int. Conf. on Evolutionary Computation*, pages 523–528. IEEE Press, 1998.

[10] J. Knowles and D. Corne. The pareto archived evolution strategy: a new baseline algorithm for multiobjective optimisation. In A. Zalzala et al., eds., *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 98–105. IEEE Press, 1999.

[11] P. Larranaga, R. Etxeberria, J. Lozano, and J. Pena. Optimization by learning and simulation of bayesian and gaussian networks. TR-EHU-KZAA-IK-4-99.

[12] M. Meila and M.I. Jordan. Estimating dependency structure as a hidden variable. In M.I. Jordan et al., eds., *Proceedings of Neural Information Processing Systems*, pages 584–590. MIT Press, 1998.

[13] H. Mühlenbein and T. Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evol. Comp.*, 7:353–376, 1999.

[14] M. Pelikan and D.E. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In M. Schoenauer et al., eds., *Parallel Problem Solving from Nature*, pages 385–394. Springer, 2000.

[15] M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. BOA: The bayesian optimization algorithm. In W. Banzhaf et al., eds., *Proc. of the 1999 Genetic and Evolutionary Computation Conference*, pages 525–532. Morgan Kaufmann, 1999.

[16] M. Pelikan, D.E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99018.ps.Z, 1999.

[17] R. Santana, A. Ochoa, and M. Soto. The mixture of trees factorized distribution algorithm. *This volume.*

[18] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.