# Expanding From Discrete To Continuous Estimation Of Distribution Algorithms: The ID$\mathbb{E}$A

Peter A.N. Bosman and Dirk Thierens

Department of Computer Science, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{*peterb*, *Dirk.Thierens*}*@cs.uu.nl*

**Abstract.** The direct application of statistics to stochastic optimization based on iterated density estimation has become more important and present in evolutionary computation over the last few years. The estimation of densities over selected samples and the sampling from the resulting distributions, is a combination of the recombination and mutation steps used in evolutionary algorithms. We introduce the framework named ID$\mathbb{E}$A to formalize this notion. By combining continuous probability theory with techniques from existing algorithms, this framework allows us to define new continuous evolutionary optimization algorithms.

## 1 Introduction

Algorithms in evolutionary optimization guide their search through statistics based on a vector of samples, often called a population. By using this stochastic information, non–deterministic induction is performed in order to attempt to use the structure of the search space and thereby aid the search for the optimal solution. In order to perform induction, these samples are combined so as to generate new solutions that will hopefully be closer to the optimum. As this process is iterated, convergence is intended to lead the algorithm to a final solution.

In the genetic algorithm [11, 14] and many variants thereof, values for problem variables are often exchanged and subsequently individually adapted. Another way of combining the samples is to regard them as being representative of some probability distribution. Estimating this probability distribution and sampling more solutions from it, is a global statistical type of inductive iterated search. Such algorithms have been proposed for discrete spaces [2–4, 12, 13, 15, 17, 19, 21], as well as in a limited way for continuous spaces [5, 10, 15, 22, 23]. An overview of this field has been given by Pelikan, Goldberg and Lobo [20].

Our goal in this paper is to apply the search for good probability density models to continuous spaces. To this end, we formalize the notion of building and using probabilistic models in a new framework named ID$\mathbb{E}$A. We show how we can adjust existing techniques to be used in the continuous case. We thereby define new evolutionary optimization algorithms. Using a set of test functions, we validate their performance.

The remainder of this paper is organized as follows. In section 2, we present the IDEA framework. In section 3 we describe a few existing algorithms that build and use probabilistic models. In section 4 we state some derivations of probability density functions (pdfs). We use the described algorithms and pdfs within the IDEA in our experiments in section 5. Topics for further research are discussed in section 6 and our final conclusions are drawn in section 7.

## 2 The IDEA

We write $\boldsymbol{a} = (\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|\boldsymbol{a}|-1})$ for a vector $\boldsymbol{a}$ of length $|\boldsymbol{a}|$. The ordering of the elements in a vector is *relevant*. We assume to have $l$ random variables available, meaning that each sample point is an $l$ dimensional vector. We introduce the notation $a\langle \boldsymbol{c} \rangle = (a_{\boldsymbol{c}_0}, a_{\boldsymbol{c}_1}, \ldots, a_{\boldsymbol{c}_{|\boldsymbol{c}|-1}})$ Let $\boldsymbol{\mathcal{L}} = (0, 1, \ldots, l-1)$ be a vector of $l$ numbers and let $\boldsymbol{\mathcal{Z}} = (Z_0, Z_1, \ldots, Z_{l-1})$ be a vector of $l$ random variables. We assume that we have an $l$ dimensional cost function $C(z\langle \boldsymbol{\mathcal{L}} \rangle)$ which without loss of generality we seek to minimize. Without any prior information on $C(z\langle \boldsymbol{\mathcal{L}} \rangle)$, we might as well assume a uniform distribution over $\boldsymbol{\mathcal{Z}}$. Now denote a probability distribution that is uniformly distributed over all $z\langle \boldsymbol{\mathcal{L}} \rangle$ with $C(z\langle \boldsymbol{\mathcal{L}} \rangle) \leq \theta$ and that has a probability of 0 otherwise, by $P^\theta(\boldsymbol{\mathcal{Z}})$. In the discrete case we have:

$$P^\theta(\boldsymbol{\mathcal{Z}})(z\langle \boldsymbol{\mathcal{L}} \rangle) = \begin{cases} \frac{1}{|\{z'\langle \boldsymbol{\mathcal{L}} \rangle | C(z'\langle \boldsymbol{\mathcal{L}} \rangle) \leq \theta\}|} & \text{if } C(z\langle \boldsymbol{\mathcal{L}} \rangle) \leq \theta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Note that if we find $P^{\theta^*}(\boldsymbol{\mathcal{Z}})$ where $\theta^* = \min_{z\langle \boldsymbol{\mathcal{L}} \rangle}\{C(z\langle \boldsymbol{\mathcal{L}} \rangle)\}$, a single sample drawn from $P^{\theta^*}(\boldsymbol{\mathcal{Z}})$ provides an optimal solution $z\langle \boldsymbol{\mathcal{L}} \rangle^*$.

A probability distribution is made up of a probability density structure (pds) and a probability density function (pdf) for each element in the pds. In graphical models literature, a pds is also called a *factorization*. Let $\boldsymbol{a} \sqcup \boldsymbol{b}$ be the splicing of $\boldsymbol{a}$ and $\boldsymbol{b}$ such that the elements of $\boldsymbol{b}$ are placed behind the elements of $\boldsymbol{a}$, giving $|\boldsymbol{a} \sqcup \boldsymbol{b}| = |\boldsymbol{a}| + |\boldsymbol{b}|$. Using graphical modelling [9], we can denote any non–clustered pds with conditional probabilities $P(Z\langle \boldsymbol{a} \rangle | Z\langle \boldsymbol{b} \rangle) = P(Z\langle \boldsymbol{a} \sqcup \boldsymbol{b} \rangle)/P(Z\langle \boldsymbol{b} \rangle)$. We let $\pi(\cdot)$ be a function that returns a vector $\pi(i) = (\pi(i)_0, \pi(i)_1, \ldots, \pi(i)_{|\pi(i)|-1})$ of indices denoting the variables that $Z_i$ is conditionally dependent on. We call the graph that results when taking the $Z_i$ as nodes and having an arc from node $Z_i$ to node $Z_j$ if and only if $i \in \pi(j)$, the pds graph. The only required condition to be able to express *any* non–clustered pds using conditional probabilities, is that this graph needs to be acyclic. By using a permutation vector $\boldsymbol{\omega}$, the definition of a pds $(\pi, \boldsymbol{\omega})$ that models conditional factorizations, can be formalized as follows:

$$P_{(\pi,\boldsymbol{\omega})}(\boldsymbol{\mathcal{Z}}) = \prod_{i=0}^{l-1} P(Z_{\boldsymbol{\omega}_i} | Z\langle \pi(\boldsymbol{\omega}_i) \rangle) \tag{2}$$

$$\text{such that } \forall_{i \in \boldsymbol{\mathcal{L}}} \langle \boldsymbol{\omega}_i \in \boldsymbol{\mathcal{L}} \wedge \forall_{k \in \boldsymbol{\mathcal{L}}-(i)} \langle \boldsymbol{\omega}_i \neq \boldsymbol{\omega}_k \rangle\rangle$$
$$\forall_{i \in \boldsymbol{\mathcal{L}}} \langle \forall_{k \in \pi(\boldsymbol{\omega}_i)} \langle k \in \{\boldsymbol{\omega}_{i+1}, \boldsymbol{\omega}_{i+2}, \ldots, \boldsymbol{\omega}_{l-1}\} \rangle\rangle$$

Each $P(Z_{\boldsymbol{\omega}_i} | Z\langle \pi(\boldsymbol{\omega}_i) \rangle)$ from equation 2 is a special case multivariate conditional pdf. This means that any such conditional pdf along with a pds $(\pi, \boldsymbol{\omega})$

defines a probability distribution over $\boldsymbol{\mathcal{Z}}$. In general, we denote a pds by $\mathfrak{f}$. The pds is constrained to be of a certain form. For instance, in the case of equation 2, the constraints impose $\mathfrak{f}$ to describe a directed acyclic graph. We denote the constrained space of all possible structures by $\mathfrak{C}$. A probability distribution over $\boldsymbol{\mathcal{Z}}$ is then formally denoted by $P_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}}), \mathfrak{f} \in \mathfrak{C}$.

Denote the largest function value of a selection of samples at iteration $t$ by $\theta_t$. We find a pds and estimate each pdf to best approximate $P^{\theta_t}(\boldsymbol{\mathcal{Z}})$. We can then sample from the resulting probability distribution to get more samples. By formalizing this rationale in an iterative algorithm, we define the *Iterated Density Estimation Evolutionary Algorithm* (I$\mathbb{DE}$A):

| $\text{I}\mathbb{DE}\text{A}(n, \tau, m, sel(), rep(), ter(), sea(), est(), sam())$ | |
|---|---|
| Initialize an empty vector of samples | $\boldsymbol{\mathcal{P}} \leftarrow ()$ |
| Add and evaluate $n$ random samples | $\boldsymbol{for}\ i \leftarrow 0\ \boldsymbol{to}\ n-1\ \boldsymbol{do}$ |
| | $\quad\boldsymbol{\mathcal{P}} \leftarrow \boldsymbol{\mathcal{P}} \sqcup \text{NewRandomVector}()$ |
| | $\quad c[\boldsymbol{\mathcal{P}}_i] \leftarrow C(\boldsymbol{\mathcal{P}}_i)$ |
| Initialize the iteration counter | $t \leftarrow 0$ |
| Iterate until termination | $\boldsymbol{while}\ \neg ter()\ \boldsymbol{do}$ |
| Select $\lfloor \tau n \rfloor$ samples | $\quad(z^0\langle\boldsymbol{\mathcal{L}}\rangle, z^1\langle\boldsymbol{\mathcal{L}}\rangle, \ldots, z^{\lfloor \tau n \rfloor - 1}\langle\boldsymbol{\mathcal{L}}\rangle) \leftarrow sel()$ |
| Set $\theta_t$ to the worst selected cost | $\quad\theta_t \leftarrow c[z^k\langle\boldsymbol{\mathcal{L}}\rangle]$ such that |
| | $\qquad\forall_{i \in \boldsymbol{\mathcal{N}_\tau}} \langle c[z^i\langle\boldsymbol{\mathcal{L}}\rangle] \leq c[z^k\langle\boldsymbol{\mathcal{L}}\rangle]\rangle$ |
| Search for a pds $\mathfrak{f}$ | $\quad\mathfrak{f} \leftarrow sea()$ |
| Estimate each pdf in $\hat{P}_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}})$ | $\quad\{\hat{P}(\cdot) \mid \hat{P}(\cdot) \Leftarrow \hat{P}_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}})\} \leftarrow est()$ |
| Create an empty vector of new samples | $\quad\boldsymbol{\mathcal{O}} \leftarrow ()$ |
| Sample $m$ new samples from $\hat{P}_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}})$ | $\quad\boldsymbol{for}\ i \leftarrow 0\ \boldsymbol{to}\ m-1\ \boldsymbol{do}$ |
| | $\qquad\boldsymbol{\mathcal{O}} \leftarrow \boldsymbol{\mathcal{O}} \sqcup sam()$ |
| Replace a part of $\boldsymbol{\mathcal{P}}$ with a part of $\boldsymbol{\mathcal{O}}$ | $\quad rep()$ |
| Evaluate the new samples in $\boldsymbol{\mathcal{P}}$ | $\quad\boldsymbol{for\ each}$ unevaluated $\boldsymbol{\mathcal{P}}_i\ \boldsymbol{do}$ |
| | $\qquad c[\boldsymbol{\mathcal{P}}_i] \leftarrow C(\boldsymbol{\mathcal{P}}_i)$ |
| Update the generation counter | $\quad t \leftarrow t + 1$ |
| Denote the required iterations by $t_{\text{end}}$ | $t_{\text{end}} \leftarrow t$ |

In the I$\mathbb{DE}$A framework, we have that $\boldsymbol{\mathcal{N}_\tau} = (0, 1, \ldots, \lfloor \tau n \rfloor - 1)$, $\tau \in [\frac{1}{n}, 1]$, $sel()$ is the selection operator, $rep()$ replaces a subset of $\boldsymbol{\mathcal{P}}$ with a subset of $\boldsymbol{\mathcal{O}}$, $ter()$ is the termination condition, $sea()$ is a pds search algorithm, $est()$ estimates each pdf and $sam()$ generates a single sample from $\hat{P}_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}})$. The notation $\hat{P}(\cdot) \Leftarrow \hat{P}_{\mathfrak{f}}$ means that $\hat{P}(\cdot)$ is one of the pdfs that is implied by the model $\mathfrak{f}$.

The I$\mathbb{DE}$A is a true evolutionary algorithm in the sense that a population of individuals is used from which individuals are selected to generate new offspring with. Using these offspring along with the parent individuals and the current population, a new population is constructed. By referring to the *iterations* in the I$\mathbb{DE}$A as *generations*, the evolutionary correspondence is even more obvious.

Note that in the I$\mathbb{DE}$A, we have used the approximation notation $\hat{P}^{\theta_t}_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}})$ instead of the true distribution $P^{\theta_t}_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}})$. An approximation is required because the determined distribution is based upon samples and the underlying density model is an assumption on the true model. This means that even though we might achieve $\hat{P}^{\theta_t}_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}}) = P^{\theta_t}_{\mathfrak{f}}(\boldsymbol{\mathcal{Z}})$, in general this is not the case.

If we set $m$ to $(n - \lfloor \tau n \rfloor)$, $sel()$ to selection by taking the best $\lfloor \tau n \rfloor$ vectors and $rep()$ to replacing the worst $(n - \lfloor \tau n \rfloor)$ vectors by the new sampled vectors, we have that $\theta_{k+1} = \theta_k - \varepsilon$ with $\varepsilon \geq 0$. This assures that the search for $\theta^*$ is conveyed through a monotonically decreasing series $\theta_0 \geq \theta_1 \geq \ldots \geq \theta_{t_{\text{end}}}$. We call an IDEA with $m$, $sel()$ and $rep()$ so chosen, a *monotonic* IDEA.

If we set $m$ in the IDEA to $n$ and set $rep()$ to replace $\boldsymbol{\mathcal{P}}$ with $\boldsymbol{\mathcal{O}}$, we obtain the EDA by Mühlenbein, Mahnig and Rodriguez [17]. In the EDA however, the threshold $\theta_t$ cannot be enforced. Note how EDA is thus an instance of IDEA.

## 3 Probability density structure search algorithms

In order to search for a pds, a metric is required that guides the search. In effect, this poses another optimization problem. The metric we use in this paper is a distance metric to the full joint pds $(\pi^+, \boldsymbol{\omega}^+)$, $\forall_{i \in \boldsymbol{\mathcal{L}}} \langle \boldsymbol{\omega}_i^+ = i \wedge \pi^+(i) = (i+1, i+2, \ldots, l-1) \rangle$. The distance metric is defined by the Kullback–Leibler (KL) divergence. We write $\boldsymbol{\mathcal{Y}}$ instead of $\boldsymbol{\mathcal{Z}}$ from now on to indicate the use of *continuous* random variables instead of either the discrete or continuous case. Using our definitions, the KL divergence can be written as [7]:

$$D(\hat{P}_{(\pi^+, \boldsymbol{\omega}^+)}(\boldsymbol{\mathcal{Y}}) || \hat{P}_{(\pi, \boldsymbol{\omega})}(\boldsymbol{\mathcal{Y}})) = -h(\hat{P}_{(\pi^+, \boldsymbol{\omega}^+)}(\boldsymbol{\mathcal{Y}})) + \sum_{i=0}^{l-1} h(\hat{P}(Y_{\boldsymbol{\omega}_i} | Y \langle \pi(\boldsymbol{\omega}_i) \rangle)) \quad (3)$$

Let $\boldsymbol{a} \sqsubseteq \boldsymbol{\mathcal{L}}, \boldsymbol{b} \sqsubseteq \boldsymbol{\mathcal{L}}$ where $\boldsymbol{a} \sqsubseteq \boldsymbol{\mathcal{L}}$ means that $\boldsymbol{a}$ contains only elements of $\boldsymbol{\mathcal{L}}$. In equation 3, $h(Y\langle \boldsymbol{a} \rangle)$ is the multivariate differential entropy and $h(Y\langle \boldsymbol{a} \rangle | Y\langle \boldsymbol{b} \rangle)$ is the conditional differential entropy. Let $dy\langle \boldsymbol{a} \rangle = \prod_{i=0}^{|\boldsymbol{a}|-1} dy_i$ be shorthand notation for the multivariate derivative. We then have:

$$h(P(Y\langle \boldsymbol{a} \rangle)) = -\int P(Y\langle \boldsymbol{a} \rangle)(y\langle \boldsymbol{a} \rangle) \ln(P(Y\langle \boldsymbol{a} \rangle)(y\langle \boldsymbol{a} \rangle)) dy\langle \boldsymbol{a} \rangle \quad (4)$$

$$h(P(Y\langle \boldsymbol{a} \rangle | Y\langle \boldsymbol{b} \rangle)) = h(P(Y\langle \boldsymbol{a} \sqcup \boldsymbol{b} \rangle)) - h(P(Y\langle \boldsymbol{b} \rangle)) \quad (5)$$

As the term $h(\hat{P}_{(\pi^+, \boldsymbol{\omega}^+)}(\boldsymbol{\mathcal{Y}}))$ in equation 3 is constant, an algorithm that searches for a pds can use the KL divergence by minimizing the sum of the conditional entropies imposed by $(\pi, \boldsymbol{\omega})$. This will cause the pds search algorithm to search for a pds as close as possible to $(\pi^+, \boldsymbol{\omega}^+)$ subject to additional constraints.

The probabilistic models used in previously proposed algorithms range from lower order structures to structures of unbounded complexity. It has been empirically shown by Bosman and Thierens [6] that a higher order pds is required to solve higher order building block problems. We shortly state three previously introduced pds search algorithms that we use in our experiments.

In the univariate distribution, all variables are regarded *independently* of each other. The PBIL by Baluja and Caruana [2], the cGA by Harik, Lobo and Goldberg [13], the UMDA by Mühlenbein and Paaß [18], and all known approaches in the continuous case prior to the IDEA [10, 22, 23], use this pds. It can be modelled by $\forall_{i \in \boldsymbol{\mathcal{L}}} \langle \pi(i) = () \wedge \boldsymbol{\omega}_i = i \rangle$, giving: $\hat{P}_{(\pi, \boldsymbol{\omega})}(\boldsymbol{\mathcal{Z}}) = \prod_{i=0}^{l-1} \hat{P}(Z_i)$.

In the MIMIC algorithm by De Bonet, Isbell and Viola [4], the pds is a chain which is constrained to $\pi(\boldsymbol{\omega}_{l-1}) = () \wedge \forall_{i \in \mathcal{L}-(l-1)} \langle \pi(\boldsymbol{\omega}_i) = (\boldsymbol{\omega}_{i+1}) \rangle$, giving $\hat{P}_{(\pi,\boldsymbol{\omega})}(\boldsymbol{\mathcal{Z}}) = (\prod_{i=0}^{l-2} \hat{P}(Z_{\boldsymbol{\omega}_i}|Z_{\boldsymbol{\omega}_{i+1}}))\hat{P}(Z_{\boldsymbol{\omega}_{l-1}})$. To find the chain, an $\mathcal{O}(l^2)$ greedy approximation algorithm is used in MIMIC to minimize the KL divergence.

If the pds is constrained so that in addition to having an acyclic pds graph, each node may have at most $\kappa$ parents, the pds is constrained to $\forall_{i \in \mathcal{L}} \langle |\pi(i)| \leq \kappa \rangle$. This general approach is used in the BOA by Pelikan, Goldberg and Cantú–Paz [19], as well as the LFDA by Mühlenbein and Mahnig [16] and the EBNA by Larrañaga, Etxeberria, Lozano and Peña. In the case of $\kappa = 1$, a polynomial time algorithm can be used to minimize the KL divergence [5]. In the case of $\kappa > 1$, a greedy algorithm is used that iteratively adds arcs to the pds graph.

There are other special case algorithms, such as the optimal dependency trees approach by Baluja and Davies [3] and the ECGA by Harik [12]. Like the LFDA, the ECGA uses minimum description length as a search metric. This metric has the advantage that the resulting pds will not be overly complex. Using the KL divergence, this can only be influenced by adjusting $\kappa$ because the KL divergence is merely a distance measure from a certain pds to $(\pi^+, \boldsymbol{\omega}^+)$. We only regard the three described $sea()$ algorithms in combination with the KL divergence metric. Note that using the KL metric and the $(\pi^+, \boldsymbol{\omega}^+)$ pds is merely an instance of the $\mathbb{IDEA}$ framework. This is also the case for using a certain pdf. The framework is to be seen separately from the algorithms that can be modelled by it.

## 4 Probability density functions

Next to the pds search algorithms from section 3, we require to specify a pdf to use. It follows from sections 2 and 3 that we require to know the multivariate differential entropy as well as the conditional pdf. In this section, we specify two well known pdfs that we use in our experiments within the $\mathbb{IDEA}$ framework.

A widely used parametric pdf is the normal pdf. Let $\boldsymbol{S} = (\boldsymbol{y}^{\boldsymbol{0}}, \boldsymbol{y}^{\boldsymbol{1}}, \ldots, \boldsymbol{y}^{|\boldsymbol{S}|-\boldsymbol{1}})$ be the set of selected samples. The sample average in dimension $j$ is then $\overline{Y_j} = \frac{1}{|\boldsymbol{S}|}\sum_{i=0}^{|\boldsymbol{S}|-1} \boldsymbol{y^i}_j$. The sample covariance matrix over variables $Y\langle \boldsymbol{a} \rangle$ is $\boldsymbol{S} = \frac{1}{|\boldsymbol{S}|}\sum_{i=0}^{|\boldsymbol{S}|-1}(y^i\langle \boldsymbol{a} \rangle - \overline{Y}\langle \boldsymbol{a} \rangle)(y^i\langle \boldsymbol{a} \rangle - \overline{Y}\langle \boldsymbol{a} \rangle)^T$. Let $s'_{ij} = \boldsymbol{S}^{-1}(i,j)$. The conditional pdf and the entropy can be stated as follows [5]:

$$f_{\mathcal{N}}(y_{\boldsymbol{a}_0}|y\langle \boldsymbol{a} - \boldsymbol{a}_0 \rangle) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y_{\boldsymbol{a}_0}-\mu)^2}{2\sigma^2}} \tag{6}$$

$$\text{where} \quad \sigma = \frac{1}{\sqrt{s'_{00}}}, \quad \mu = \frac{\overline{Y_{\boldsymbol{a}_0}} s'_{00} - \sum_{i=1}^{|\boldsymbol{a}|-1}(y_{\boldsymbol{a}_i} - \overline{Y_{\boldsymbol{a}_i}})s'_{i0}}{s'_{00}}$$

$$h(Y\langle \boldsymbol{a} \rangle) = \frac{1}{2}(|\boldsymbol{a}| + \ln((2\pi)^{|\boldsymbol{a}|}\det(\boldsymbol{S}))) \tag{7}$$

The non–parametric normal kernels pdf places a normal pdf over every available sample point. Let $\mathfrak{s}_i$ be a fixed standard deviation in the $i$-th dimension. The conditional pdf and the entropy can then be stated as follows [8]:

$$f_{\mathcal{N}_K}(y_{\boldsymbol{a}_0}|y\langle \boldsymbol{a}-\boldsymbol{a}_0\rangle) = \sum_{i=0}^{|\boldsymbol{S}|-1} \boldsymbol{\nu}_i \frac{1}{\mathfrak{s}_{\boldsymbol{a}_0}\sqrt{2\pi}} e^{\frac{-(y_{\boldsymbol{a}_0} - \boldsymbol{y}^{\boldsymbol{i}}{}_{\boldsymbol{a}_0})^2}{2\mathfrak{s}^2_{\boldsymbol{a}_0}}} \tag{8}$$

$$\text{where } \boldsymbol{\nu}_i = \frac{e^{-\sum_{j=1}^{|\boldsymbol{a}|-1} \frac{(y_{\boldsymbol{a}_j} - \boldsymbol{y}^{\boldsymbol{i}}{}_{\boldsymbol{a}_j})^2}{2\mathfrak{s}^2_{\boldsymbol{a}_j}}}}{\sum_{k=0}^{|\boldsymbol{S}|-1} e^{-\sum_{j=1}^{|\boldsymbol{a}|-1} \frac{(y_{\boldsymbol{a}_j} - \boldsymbol{y}^{\boldsymbol{k}}{}_{\boldsymbol{a}_j})^2}{2\mathfrak{s}^2_{\boldsymbol{a}_j}}}}$$

$$h(Y\langle \boldsymbol{a}\rangle) = \tag{9}$$

$$\frac{1}{2}\ln\left(|\boldsymbol{S}|^2(2\pi)^{|\boldsymbol{a}|}\prod_{j=0}^{|\boldsymbol{a}|-1}\mathfrak{s}^2_{\boldsymbol{a}_j}\right) - \int f_{\mathcal{N}_K}(y\langle \boldsymbol{a}\rangle)\ln\left(\sum_{i=0}^{|\boldsymbol{S}|-1} e^{-\sum_{j=0}^{|\boldsymbol{a}|-1}\frac{(y_{\boldsymbol{a}_j} - \boldsymbol{y}^{\boldsymbol{i}}{}_{\boldsymbol{a}_j})^2}{2\mathfrak{s}^2_{\boldsymbol{a}_j}}}\right) dy\langle \boldsymbol{a}\rangle$$

An alternative pdf to the two described above, is the histogram pdf. Using this pdf however does not scale up very well [7] and leads to an exponential iteration running time in the order of $r^{\kappa}$ where $r$ is the amount of bins to use in each dimension. The normal pdf is very efficient but very cluster insensitive. The normal kernels pdf is very sensitive to clusters but may very quickly overfit the data. In addition, the running time each iteration for using the latter pdf tends to be a lot greater than for the normal pdf.

## 5 Experiments

We have used the following continuous function optimization problems:

| | | |
|---|---|---|
| $C_1$ | $\frac{1}{4000}\sum_{i=0}^{l-1}(y_i-100)^2 - \prod_{i=0}^{l-1}\cos(\frac{y_i-100}{\sqrt{i+1}})+1$ | $[-600,600]^l$ |
| $C_2$ | $\gamma_i = \frac{24}{1000}(i+2) - y_i$ | $[-3,3]^l$ |
| $C_3$ | $\gamma_0 = Y_0, \gamma_i = y_i + \gamma_{i-1}$ | $[-3,3]^l$ |
| $C_4$ | $\gamma_0 = Y_0, \gamma_i = y_i + \sin(\gamma_{i-1})$ | $[-3,3]^l$ |

Function $C_1$ is Griewank's function and $C_2$, $C_3$ and $C_4$ are test functions by Baluja. For the latter three functions we have to *maximize* $100/(10^{-5} + \sum_{i=0}^{l-1}|\gamma_i|)$. Griewank's function should be *minimized*.

We use *monotonic* I$\mathbb{DE}$As with the normal pdf and the normal kernels pdf. Furthermore, we use the KL metric and truncation selection. The amount of available samples $\lfloor \tau n \rfloor$ strongly influences the effectiveness of density estimation. We expect a better performance if this amount goes up. Therefore, we fix $\tau$ and increase $n$. To be more precise, we use the rule of thumb by Mühlenbein and Mahnig [16] for FDA and set $\tau$ to 0.3. If all of the solutions differed by less than $5 \cdot 10^{-7}$, termination was enforced. The $\mathfrak{s}_i$ standard deviation parameters for the normal kernels pdf were determined as $(\alpha \cdot \text{range}^i)/\lfloor \tau n \rfloor$ with $\alpha = 3$ for $C_1$.

Our results are presented using the notion of *relative function evaluations* RFE $= n_e$RT, where $n_e$ is the required amount of function evaluations. Let

FT($x$) be the time spent to perform $x$ random function evaluations and let TT be the average total algorithm time spent including the $n_e$ function evaluations. The *relative time* is defined as $\mathrm{RT} = (\mathrm{TT} - \mathrm{FT}(n_e))/\mathrm{FT}(n_e)$. The RFE metric is a cpu independent fair running time comparison.

Functions $C_1$ and $C_2$ can be optimized efficiently by determining a value for each variable separately. This is not the case for functions $C_3$ and $C_4$. We therefore only used the univariate distribution on $C_1$. In figure 1, the scalability on $C_1$ and $C_3$ is shown, computed over 20 runs. We only used the normal pdf on $C_3$. The RFE at the minimal value of $n$ at which $C_1$ was minimized to a value to reach (VTR) of $10^{-6}$ in all runs is shown on a linear scale. The computation time scales approximately linearly for the ID$\mathbb{E}$A variants. For $l \in \{250, 300\}$, the VTR was never reached for the normal kernels. By allowing $\alpha$ to vary adaptively, thereby obtaining a more flexible density estimator, this might be overcome. For $C_3$, we used a VTR of 5. The results are shown on a logarithmic scale. The true pds of $C_3$ can be seen to be the full joint distribution. Using this pds scales up polynomially, whereas using the univariate model scales up exponentially. Using the graph search with $\kappa = 1$ seems to scale up polynomially, but tests for a larger value of $l$ need to be run to be sure.

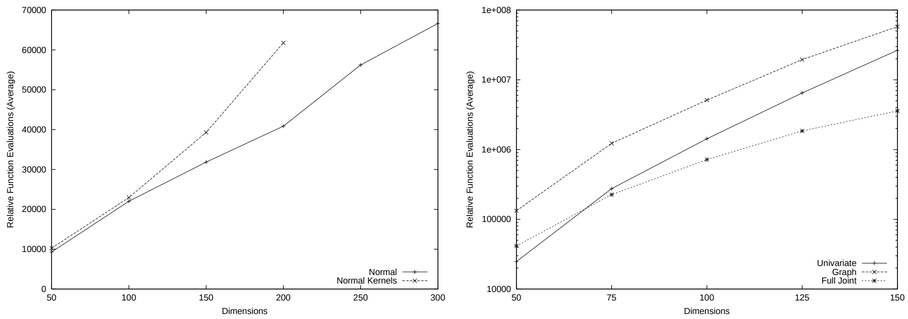


**Fig. 1.** Results on $C_1$ (left, linear) and $C_3$ (right, logarithmic) for increasing dimension.

We compared the ID$\mathbb{E}$A using the normal pdf to Evolution Strategies [1] (ES) on $C_1$. The ES has a $(\mu, \lambda)$ strategy with $\lambda = 7\mu$ and independent mutations using either individual standard deviations $n_\sigma = l$ or a single standard deviation $n_\sigma = 1$. We initialized the standard deviations to 3.0 and used $\tau_{\mathrm{G}} = 1/\sqrt{2l}$ and $\tau_i = 1/\sqrt{2\sqrt{l}}$. Table 2 shows the success rate (SR), which is the relative amount of times the VTR of $10^{-6}$ was reached. The results are computed over 10 runs. The parameters for the ES are indicated by $(n_\sigma, \mu, \lambda)$ and for the ID$\mathbb{E}$A by $(n, \tau)$. We allowed $10^5$ evaluations if $l = 10$ and $10^6$ evaluations if $l = 300$. In all cases except for ES $(l, 30, 210)$ for $l = 300$ and ID$\mathbb{E}$A $(500, 0.3)$ for $l \in \{10, 300\}$, premature convergence resulted in SR $< 100\%$. In the other two cases, the maximum of evaluations was reached before convergence in unsuccessful runs.

The continuous PBIL approach by Sebag and Ducoulombier [22] was tested on functions $C_2$, $C_3$ and $C_4$. We used ID$\mathbb{E}$As with the KL metric and the normal pdf in combination with the univariate distribution, the chain search algorithm, the exact graph search algorithm for $\kappa = 1$ and a fixed chain according to the

| | ES | | | | IDEA | |
|---|---|---|---|---|---|---|
| $l$ | $(1, 15, 105)$ | $(l, 15, 105)$ | $(1, 30, 210)$ | $(l, 30, 210)$ | $(250, 0.3)$ | $(500, 0.3)$ |
| 10 | 20% | 10% | 20% | 40% | 100% | 70% |
| 300 | 70% | 0% | 60% | 0% | 0% | 100% |

**Fig. 2.** Success rates on $C_1$ in 10 and 300 dimensions.

function definitions, so $\boldsymbol{\omega}_i = l - i - 1$, $\pi(0) = ()$ and $\pi(i) = i - 1$, $i \geq 1$. For a given pdf, results obtained with this pds is an upper bound for any pds in which each node may have at most a single parent. We also tested the full joint pds. We have used $l = 100$, a maximum of $2 \cdot 10^5$ evaluations and we have averaged the results over 20 runs. We increased $n$ in steps of 25 to find the best value for $n$. Repeating earlier reported results [22], table 3 indicates that our approaches perform better. We note that using the full joint pds requires very large values for $n$. As a result, the amount of generations is very small since we are allowed only $2 \cdot 10^5$ evaluations. This strongly influences the results.

| | $C_2$, $l = 100$ | | | $C_3$, $l = 100$ | | | $C_4$, $l = 100$ | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | $\overline{C_2}$ | $n$ | RT | $\overline{C_3}$ | $n$ | RT | $\overline{C_4}$ | $n$ | RT |
| $(10+50)$–ES | 399.07 | — | — | 2.91 | — | — | 7.56 | — | — |
| PBIL (Binary) | 16.43 | — | — | 2.12 | — | — | 4.4 | — | — |
| PBIL (Gray) | 366,77 | — | — | 2.62 | — | — | 5.61 | — | — |
| $\text{PBIL}_C$ | 4803 | — | — | 4.76 | — | — | 11.18 | — | — |
| IDEA $\text{No}_\text{U}$ | 9999999.87 | 225 | 4.92 | 4.51 | 150 | 5.00 | 13.40 | 250 | 0.95 |
| IDEA $\text{No}_\text{C}$ | 9999999.90 | 250 | 17.83 | 5.30 | 200 | 18.65 | 14.85 | 300 | 3.71 |
| IDEA $\text{No}_\text{G}$ | **9999999.96** | **350** | **130.17** | **7.50** | **275** | **44.32** | **27.73** | **550** | **4.95** |
| IDEA $\text{No}_\text{FC}$ | 9999999.88 | 350 | 5.68 | 13.48 | 350 | 6.03 | 49.65 | 450 | 1.07 |
| IDEA $\text{No}_\text{J}$ | 1.81 | 7000 | 33.13 | 347.00 | 3575 | 59.76 | 360.62 | 3375 | 11.91 |

**Fig. 3.** Results on $C_2$, $C_3$ and $C_4$ in 100 dimensions.

## 6 Discussion

Selecting what value of $\alpha$ to use for the normal kernels pdf is dependent on the optimization problem. The value of $\alpha$ determines the smoothness of the fit [7], which makes it intuitive that increasing $\alpha$ on smooth functions should give better results. In our experiments, we empirically determined $\alpha$, but it is worthwhile to investigate how to use $\alpha$ adaptively.

Using the normal kernels pdf quickly tends to overfit the sample vector. This can somewhat be regulated by $\alpha$, but not entirely. The tendency to overfit became apparent as the approach was highly sensitive to the value of $\lfloor \tau n \rfloor$. Using the normal pdf on the other hand almost always underfits a sample vector. The normal mixture pdf with regularization is therefore an interesting trade–off that seems very worthwhile to investigate next for multiple reasons.

In this paper, we have used the KL metric. The drawback of this metric is that it is merely a distance metric to the fully joint distribution. This means that unless the additional constraints on the pds are very specific, a pds search algorithm will most likely not result in using the problem structure in an effective way. Therefore, metrics such as the minimum description length or other direct conditionality tests are strongly worth investigating. The algorithms tested in this paper should be seen independently from the IDEA framework.

When we increase the size $l$ of a problem, we are estimating a probability distribution in a highly dimensional space. Using the full joint pds in that case poses problems for an IDEA. Estimating joint pdfs in highly dimensional spaces can require a large amount of time as well as many samples to justify the estimation because of the curse of dimensionality. However, the assumption that the cost function is built up of bounded lower order interactions between the problem variables is usually made. This implies that the actual pdfs that are being estimated, are of a lower order. If the optimization problem is not built up of such bounded lower order building blocks, using IDEAs is potentially a non–scalable approach, depending on the pdf that is used and the problem definition.

## 7 Conclusions

We have used the algorithmic framework IDEA for modelling iterated density estimation evolutionary algorithms. These algorithms make use of density estimation techniques to build a probability distribution over the variables that code a problem in order to perform optimization. To this end, a probability density structure must be found and subsequently be used in density estimation. For a set of existing search algorithms, we have applied and tested them in the IDEA framework using two different density estimation models.

The experiments indicate that building and using probabilistic models for continuous optimization problems is promising. This, in combination with its modelling capabilities, shows that the IDEA is general, applicable and effective.

## References


1. T. Bäck and H-P. Schwefel. Evolution strategies I: Variants and their computational implementation. In G. Winter, J. Priaux, M. Galn, and P. Cuesta, editors, *Genetic Algorithms in Engineering and Computer Science, Proc. of the First Short Course EUROGEN'95*, pages 111–126. Wiley, 1995.
2. S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russell, editors, *Proc. of the 12th Int. Conf. on Machine Learning*, pages 38–46. Morgan Kauffman Pub., 1995.
3. S. Baluja and S. Davies. Using optimal dependency–trees for combinatorial optimization: Learning the structure of the search space. In D.H. Fisher, editor, *Proc. of the 1997 Int. Conf. on Machine Learning*. Morgan Kauffman Pub., 1997.
4. J.S. De Bonet, C. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing*, 9, 1996.
5. P.A.N. Bosman and D. Thierens. An algorithmic framework for density estimation based evolutionary algorithms. Utrecht University Technical Report UU–CS–1999–46. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-1999/1999-46.ps.gz, 1999.
6. P.A.N. Bosman and D. Thierens. Linkage information processing in distribution estimation algorithms. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon,

V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proc. of the GECCO–1999 Genetic and Evolutionary Computation Conference*, pages 60–67. M.K. Pub., 1999.
7. P.A.N. Bosman and D. Thierens. Continuous iterated density estimation evolutionary algorithms within the IDEA framework. Utrecht Univ. Tech. Rep. UU–CS–2000–15. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2000/2000-15.ps.gz, 2000.
8. P.A.N. Bosman and D. Thierens. IDEAs based on the normal kernels probability density function. Utrecht University Technical Report UU–CS–2000–11. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2000/2000-11.ps.gz, 2000.
9. D. Edwards. *Introduction To Graphical Modelling*. Springer–Verlag, 1995.
10. M. Gallagher, M. Fream, and T. Downs. Real–valued evolutionary optimization using a flexible probability density estimator. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proc. of the GECCO–1999 Gen. and Evol. Comp. Conf.*, pages 840–846. M.K. Pub., 1999.
11. D.E. Goldberg. *Genetic Algorithms In Search, Optimization, And Machine Learning*. Addison–Wesley, Reading, 1989.
12. G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Tech. Report 99010. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99010.ps.Z, 1999.
13. G. Harik, F. Lobo, and D.E. Goldberg. The compact genetic algorithm. In *Proc. of the 1998 IEEE Int. Conf. on Evol. Comp.*, pages 523–528. IEEE Press, 1998.
14. J.H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.
15. P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña. Optimization by learning and simulation of bayesian and gaussian networks. University of the Basque Country Technical Report EHU-KZAA-IK-4/99. http://www.sc.ehu.es/ccwbayes/postscript/kzaa-ik-04-99.ps, 1999.
16. H. Mühlenbein and T. Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evol. Comp.*, 7:353–376, 1999.
17. H. Mühlenbein, T. Mahnig, and O. Rodriguez. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5:215–247, 1999.
18. H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. binary parameters. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 178–187. Springer, 1998.
19. M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. BOA: The bayesian optimization algorithm. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proc. of the GECCO–1999 Genetic and Evolutionary Computation Conference*, pages 525–532. Morgan Kaufmann Pub., 1999.
20. M. Pelikan, D.E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. IlliGAL Technical Report 99018. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99018.ps.Z, 1999.
21. M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, K. Chawdry, and K. Pravir, editors, *Advances in Soft Computing – Engineering Design and Manufacturing*. Springer–Verlag, 1999.
22. M. Sebag and A. Ducoulombier. Extending population–based incremental learning to continuous search spaces. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 418–427. Springer, 1998.
23. I. Servet, L. Trave-Massuyes, and D. Stern. Telephone network traffic overloading diagnosis and evolutionary computation technique. In J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Proc. of Artificial Evolution '97*, pages 137–144. Springer Verlag, LNCS 1363, 1997.