Continuous Iterated Density Estimation Evolutionary Algorithms Within The IDEA Framework

(Full (Technical Report) Version)

Peter A.N. Bosman
peterb@cs.uu.nl

Dirk Thierens
Dirk.Thierens@cs.uu.nl

Department of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

In this paper, we formalize the notion of performing optimization by iterated density estimation evolutionary algorithms as the IDEA framework. These algorithms build probabilistic models and estimate probability densities based upon a selection of available points. We show how these probabilistic models can be built and used for different probability density functions within the IDEA framework. We put the emphasis on techniques for vectors of continuous random variables and thereby introduce new continuous evolutionary optimization algorithms.

1 Introduction

Genetic algorithms (GAs) [11, 14] and many variants thereof combine the material within a subset of the solutions. Often this is done by exchanging values for problem variables, followed by individual adaptation of these values. Another approach is to identify a subset of the solutions 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. Algorithms of this sort have been proposed using different types of probabilistic models for discrete spaces [1, 2, 3, 12, 13, 17, 19, 21], as well as in a limited way for continuous spaces [4, 10, 22, 23]. An overview of this field has been given by Pelikan, Goldberg and Lobo [20].

Our goal in this paper is to formalize the notion of building and using probabilistic models in evolutionary optimization algorithms and to apply the search for good probability density models to continuous spaces. Mühlenbein, Mahnig and Rodriguez [17] first presented a general framework for this type of algorithm, named EDA (Estimation of Distribution Algorithm). In this paper, we make certain steps of the EDA more explicit within a new framework, named IDEA (Iterated Density Estimation Evolutionary Algorithm). We do not introduce any new way of linkage information processing or other means of exploiting problem structure, but show how we can adjust the existing techniques to be used in the continuous case. By specifying earlier proposed search algorithms within the new framework along with certain derivations of a few probability density functions, we define new evolutionary optimization algorithms. Using a set of test functions, we validate their performance. So far, we are able to give better results on a set of hard continuous function optimization problems than other approaches.

The remainder of this paper is organized as follows. In section 2, we provide a background in probability theory and introduce some notation. In section 3, we formalize the IDEA framework. Subsequently, in sections 4 and 5, algorithms to build probabilistic models and derivations of probability density functions to use within the framework are given. Our experiments are presented in section 6. Topics for further research are discussed in section 7 and our final conclusions are drawn in section 8.

2 Probability theory background and notation

In probability theory, a classic distinction is made between the discrete and the continuous case. We shall restrict this introduction to the continuous case. We write discrete random variables as X_i and continuous random variables as Y_i. When we do not distinguish between the continuous and the discrete case, we use random variables Z_i. We write vectors of random variables as Z or Z^j.

Let $\mathcal{L} = \{0, 1, \ldots, l-1\}$, $d_A = d\{y_i | i \in A\}$ and $\boldsymbol{f}(A) = f_{0,1,\ldots,|A|-1}(y_0, y_1, \ldots, y_{|A|-1})$. The multivariate joint probability density function (pdf) $\boldsymbol{f}(\mathcal{L})$ for l continuous random variables $Y = (Y_0, Y_1, \ldots, Y_{l-1})$ can be written as:

$$\int_{a_0}^{b_0}\int_{a_1}^{b_1}\cdots\int_{a_{l-1}}^{b_{l-1}} \boldsymbol{f}(\mathcal{L})d_{\mathcal{L}} = P(Y \in A) \tag{1}$$

$$\text{such that } \int \boldsymbol{f}(\mathcal{L})d_{\mathcal{L}} = 1, \quad \boldsymbol{f}(\cdot) \geq 0, \text{ and}$$

$$A = \left(\prod_{i=0}^{l-1}[a_i, b_i]\right) \subseteq \mathbb{R}^l$$

We write $P(Y_i)$ for f_i, making $P(Y_i)$ a (density) function. In the discrete case, we write $P(X_i)(k) = P(X_i = k)$. Estimating the joint distribution over $Z = (Z_0, Z_1, \ldots, Z_{l-1})$ of given sample points, can be done by specifying a pdf as in equation 1. However, we may also regard subsets of variables in a non–joint fashion such as $P'(Z) = P(Z_1 Z_9)P(\{Z_j | j \in \mathcal{L} - \{1, 9\}\})$. A complete probability density $P(Z)$ is therefore defined by its probability density structure (pds) as well as a pdf for each element of this pds. In graphical models literature, a pds is also called a *factorization*. The definition of conditional probability is:

$$P(\{Y_j | j \in A\} | \{Y_j | j \in B\}) = \frac{P(\{Y_j | j \in A \cup B\})}{P(\{Y_j | j \in B\})} \tag{2}$$

We define $\pi(\cdot)$ to be a function that returns a vector $\pi(i) = (\pi(i)_0, \pi(i)_1, \ldots, \pi(i)_{|\pi(i)|-1})$ of parent variable indices and let $\boldsymbol{\omega} = (\omega_0, \omega_1, \ldots, \omega_{l-1})$be a vector of ordering variable indices. We write $\hat{P}(\cdot)$ as an approximation to the true density $P(\cdot)$. Using equation 2, the pds can be uniquely specified by a pair $(\pi, \boldsymbol{\omega})$:

$$\hat{P}_{\pi,\boldsymbol{\omega}}(Z) = \prod_{i=0}^{l-1} \hat{P}(Z_{\omega_i} | \{Z_j | j \in \pi(\omega_i)\}) \tag{3}$$

$$\text{such that } \forall_{i \in \mathcal{L}} \langle \omega_i \in \mathcal{L} \wedge \forall_{k \in \mathcal{L}-\{i\}} \langle \omega_i \neq \omega_k \rangle \rangle$$

$$\forall_{i \in \mathcal{L}} \langle \forall_{k \in \pi(\omega_i)} \langle k \in \{\omega_{i+1}, \omega_{i+2}, \ldots, \omega_{l-1}\} \rangle \rangle$$

The above constraints enforce that there are no cyclic dependencies. Scanning the variables in the order $Y_{\omega_{l-1}}, Y_{\omega_{l-2}}, \ldots, Y_{\omega_0}$ ensures that the parent variables that a variable is conditioned on, will already have been regarded. The pds of $P(Y_0, Y_2, Y_3)P(Y_1|Y_0) = P(Y_1|Y_0)P(Y_0|Y_2, Y_3)P(Y_2|Y_3)P(Y_3)$ can for example be specified as $\omega_0 = 1$, $\omega_1 = 0$, $\omega_2 = 2$, $\omega_3 = 3$, $\pi(0) = (2, 3)$, $\pi(1) = (0)$, $\pi(2) = (3)$, $\pi(3) = \emptyset$.

A well known distance metric from $\hat{P}_{\pi,\boldsymbol{\omega}}(Y)$ to $\hat{P}_{\pi',\boldsymbol{\omega}'}(Y)$ is the Kullback–Leibler (KL) divergence, which is also called relative entropy [15]:

$$D(\hat{P}_{\pi,\boldsymbol{\omega}}(Y) || \hat{P}_{\pi',\boldsymbol{\omega}'}(Y)) = \int \hat{P}_{\pi,\boldsymbol{\omega}}(Y) \ln\left(\frac{\hat{P}_{\pi,\boldsymbol{\omega}}(Y)}{\hat{P}_{\pi',\boldsymbol{\omega}'}(Y)}\right) d_{\mathcal{L}} \tag{4}$$

The full joint probability distribution is modelled by the pds $(\pi^+, \boldsymbol{\omega}^+)$ that satisfies $\forall_{i \in \mathcal{L}} \langle \omega_i^+ = i \wedge \pi^+(i) = (i+1, i+2, \ldots, l-1) \rangle$, implying $\hat{P}(Y) = \hat{P}_{\pi^+,\boldsymbol{\omega}^+}(Y)$. Let $\mathcal{S} \subseteq \mathcal{L}$. Using $\int \hat{P}(Y) \ln(\hat{P}(\{Y_j | j \in \mathcal{S}\}))d_{\mathcal{L}} = \int(\int \hat{P}(Y) d_{\mathcal{L}-\mathcal{S}}) \ln(\hat{P}(\{Y_j | j \in \mathcal{S}\}))d_{\mathcal{S}} = \int(\hat{P}(\{Y_j | j \in \mathcal{S}\}) \ln(\hat{P}(\{Y_j | j \in \mathcal{S}\}))d_{\mathcal{S}}$, the KL divergence from any pds $(\pi, \boldsymbol{\omega})$ to $(\pi^+, \boldsymbol{\omega}^+)$, can be written as follows:

$$D(\hat{P}_{\pi^+,\boldsymbol{\omega}^+}(Y) || \hat{P}_{\pi,\boldsymbol{\omega}}(Y)) = \tag{5}$$

$$\int \hat{P}(Y) \ln(\hat{P}(Y)) - \hat{P}(Y) \ln(\hat{P}_{\pi,\boldsymbol{\omega}}(Y)) d_{\mathcal{L}} =$$

$$\int \hat{P}(Y) \ln(\hat{P}(Y)) d_{\mathcal{L}} - \int \hat{P}(Y) \ln(\hat{P}_{\pi,\boldsymbol{\omega}}(Y)) d_{\mathcal{L}} =$$

$$-h(Y) - \sum_{i=0}^{l-1} \int \hat{P}(Y) \ln(\hat{P}(Y_{\omega_i} | \{Y_j | j \in \pi(\omega_i)\})) d_{\mathcal{L}} =$$

$$-h(Y) - \sum_{i=0}^{l-1} \left(\int \hat{P}(Y) \ln(\hat{P}(\{Y_j | j = \omega_i \vee j \in \pi(\omega_i)\})) d_{\mathcal{L}} - \int \hat{P}(Y) \ln(\hat{P}(\{Y_j | j \in \pi(\omega_i)\})) d_{\mathcal{L}} \right) =$$

$$-h(Y) + \sum_{i=0}^{l-1} h(Y_{\omega_i} | \{Y_j | j \in \pi(\omega_i)\})$$

In equation 5, $h(\{Y_j | j \in \mathcal{S}\})$ stands for the multivariate differential entropy and $h(\{Y_j | j \in A\} | \{Y_j | j \in B\})$ stands for the conditional differential entropy, which were defined by Shannon [24]:

$$h(\{Y_j | j \in \mathcal{S}\}) = -\int \boldsymbol{f}(\mathcal{S}) \ln(\boldsymbol{f}(\mathcal{S})) d_{\mathcal{S}} \tag{6}$$

$$h(\{Y_j | j \in A\} | \{Y_j | j \in B\}) = h(\{Y_j | j \in A \cup B\}) - h(\{Y_j | j \in B\}) \tag{7}$$

As the expression $h(Y)$ in equation 5 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.

3 The IDEA framework

Assume we have a function optimization problem with cost function $C(Z)$ which without loss of generality we seek to minimize. Let $P^{\theta}(Z)$ denote a probability distribution that is uniform over all vectors Z with $C(Z) \leq \theta$ and equals 0 for all other vectors. In the discrete case, $P^{\theta}(Z)$ can be denoted by:

$$P^{\theta}(X) = \begin{cases} \frac{1}{|\{X' | C(X') \leq \theta\}|} & \text{if } C(X) \leq \theta \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Sampling from $P^{\theta}(Z)$ gives more points Z' with $C(Z') \leq \theta$. Moreover, a single sample from $P^{\theta^*}(Z)$ with $\theta^* = \min_Z\{C(Z)\}$, gives us an optimal solution vector Z^*. This rationale was first posed by De Bonet, Isbell and Viola [3] and has been formalized in the *Iterated Density Estimation Evolutionary Algorithm* (IDEA) by Bosman and Thierens [4]:

IDEA$(n, \tau, m, sel(), rep(), ter(), sea(), est(), sam())$
1 Initialize an empty set of samples
$P \leftarrow \emptyset$
2 Add n random samples and evaluate them
for $i \leftarrow 0$ ***to*** $n-1$ ***do***
2.1 $P \leftarrow P \cup$ NewRandomVector(Z^i)
2.2 $c[i] \leftarrow C(Z^i)$
3 Initialize the iteration counter
$t \leftarrow 0$
4 Iterate estimating densities and sampling
while $\neg ter()$ ***do***
4.1 Select $\lfloor \tau n \rfloor$ samples
$\{Z^{(S)i} \mid i \in \mathcal{N}_\tau\} \leftarrow sel()$
4.2 Set θ_t to the worst selected sample cost
$\theta_t \leftarrow c[Z^{(S)k}]$ such that
$\forall_{i \in \mathcal{N}_\tau} \langle c[Z^{(S)i}] \leq c[Z^{(S)k}] \rangle$
4.3 Search for a pds
$(\pi, \boldsymbol{\omega}) \leftarrow sea()$
4.4 Estimate the density functions
$\{\hat{P}(Z_{\omega_i} | \{Z_j | j \in \pi(\omega_i)\}) | i \in \mathcal{L}\} \leftarrow est()$
4.5 Initialize an empty set of new samples
$O \leftarrow \emptyset$
4.6 Sample m new samples from $\hat{P}(\cdot)$
for $i \leftarrow 0$ ***to*** $m-1$ ***do***
4.6.1 $O \leftarrow O \cup sam()$
4.7 Replace a subset of P with a subset of O
$rep()$
4.8 Evaluate the new samples in P
for each new $Z^i \in P$ ***do***
4.8.1 $c[i] \leftarrow C(Z^i)$
4.9 Update the generation counter
$t \leftarrow t+1$
5 Denote the required iterations by t_{end}
$t_{\text{end}} \leftarrow t$

In the IDEA framework, we have that $\mathcal{N}_\tau = \{0, 1, \ldots, \lfloor \tau n \rfloor - 1\}$, $\tau \in [\frac{1}{n}, 1]$, $sel()$ is the selection operator, $rep()$ replaces a subset of P with a subset of O, $ter()$ is the termination condition, $sea()$ is a pds search algorithm, $est()$ estimates the density functions and $sam()$ generates a single sample using the estimated densities.

The IDEA 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 IDEA as *generations*, the evolutionary characteristic is even more obvious.

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

If we set m to $(n - \lfloor \tau n \rfloor)$, $sel()$ to taking the best $\lfloor \tau n \rfloor$ vectors and $rep()$ to replacing the worst $(n - \lfloor \tau n \rfloor)$ vectors by the new samples, we have that $\theta_{k+1} = \theta_k - \varepsilon$ with $\varepsilon \geq 0$. This assures that the search for θ^* 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 P completely with O, we obtain the EDA by Mühlenbein, Mahnig and Rodriguez [17]. In the EDA however, the probability plateau θ_t cannot be enforced. Note how EDA is thus an instance of IDEA.

In order to define algorithms in the IDEA framework, we require to have a pds search algorithm $sea()$. Algorithm $est()$ will then estimate the probability density functions and algorithm $sam()$ will use them to sample a new point. When l goes up, using the full joint pds will pose problems for an IDEA. Estimating probability densities in highly dimensional spaces can require a large amount of time as well as many samples to justify the estimation. Therefore, the assumption that the cost function is built up of bounded lower order interactions between the problem variables is usually made. Such an assumption justifies using a $sea()$ algorithm that searches for a pds subject to constraints such as $\forall_{i \in \mathcal{L}} \langle |\pi(i)| \leq \kappa \rangle$.

4 Probability density structure search algorithms

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 [5] that structures of a greater complexity that allow for interactions between multiple variables, are indeed required to solve higher order building block problems. We shortly go over a few of these algorithms and determine the pds search method that is used.

The PBIL by Baluja and Caruana [1], 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 IDEA [10, 22, 23], use the univariate distribution. This pds causes the IDEA to process the variables independently of each other. Constraining the pds to this structure, it can be modelled by $\forall_{i\in\mathcal{L}}\langle\pi(i)=\emptyset\wedge\omega_i=i\rangle$, giving:

$$\hat{P}(Z)=\prod_{i=0}^{l-1}\hat{P}(Z_i) \tag{9}$$

In the approach by De Bonet, Isbell and Viola [3], known as MIMIC, the pds is a chain. In addition to the constraints from equation 3, this imposes the constraints $\pi(\omega_{l-1})=\emptyset\wedge\forall_{i\in\mathcal{L}-\{l-1\}}\langle\pi(\omega_i)=(\omega_{i+1})\rangle$, giving:

$$\hat{P}(Z)=\left(\prod_{i=0}^{l-2}\hat{P}(Z_{\omega_i}|Z_{\omega_{i+1}})\right)\hat{P}(Z_{\omega_{l-1}}) \tag{10}$$

The KL divergence from this pds to $(\pi^+,\boldsymbol{\omega}^+)$ is minimized by minimizing $(\sum_{i=0}^{l-2}h(Z_{\omega_i}|Z_{\omega_{i+1}}))+h(Z_{\omega_{l-1}})$ over all feasible $(\pi,\boldsymbol{\omega})$. The search algorithm in MIMIC, which we refer to as chain–search, first finds $Z_{\omega_{l-1}}$ such that $h(Z_{\omega_{l-1}})$ is minimal. Then, it iteratively selects Z_{ω_i} for decreasing i, such that $h(Z_{\omega_i}|Z_{\omega_{i+1}})$ is minimal. This greedy approximation algorithm runs in $\mathcal{O}(l^2)$ time.

If the chain constraints are relaxed to tree constraints, the KL divergence can be minimized in $\mathcal{O}(l^2)$ time using an algorithm by Chow and Liu [7], whereas the chain–search is approximate. This is the approach by Baluja and Davies [2]. The algorithm, which we refer to as tree–search, first randomly selects a root $Z_{\omega_{l-1}}$ and sets it as the parent of all other variables. It then iteratively selects Z_{ω_i} for decreasing i, such that $h(Z_{\omega_i})-h(Z_{\omega_i}|Z_{parent[\omega_i]})$ is maximal. The parent of each Z_j, $j\in\mathcal{L}-\{\omega_i,\omega_{i+1},\ldots,\omega_{l-1}\}$, is then set to $arg\max_{k\in\{parent[j],\omega_i\}}\{h(Z_j)-h(Z_j|Z_k)\}$. The tree pds imposes the additional constraints $\pi(\omega_{l-1})=\emptyset\wedge\forall_{i\in\mathcal{L}-\{l-1\}}\langle|\pi(\omega_i)|=1\wedge\pi(\omega_i)_0\in\{\omega_{i+1},\omega_{i+2},\ldots,\omega_{l-1}\}\rangle$, giving:

$$\hat{P}(Z)=\left(\prod_{i=0}^{l-2}\hat{P}(Z_{\omega_i}|Z_{\pi(\omega_i)_0})\right)\hat{P}(Z_{\omega_{l-1}}) \tag{11}$$

If the tree constraints are relaxed further to directed acyclic graph (dag) constraints, the only additional constraint is $\forall_{i\in\mathcal{L}}\langle|\pi(i)|\leq\kappa\rangle$, giving equation 3. Minimizing the KL divergence then comes down to minimizing the sum of conditional entropies from equation 5. The BMDA by Pelikan and Mühlenbein [21] uses the special case pds with $\kappa=1$, as does the BOA by Pelikan, Goldberg and Cantú–Paz [19]. In addition, a search algorithm for the case of $\kappa>1$ is also incorporated in the BOA. In the LFDA by Mühlenbein and Mahnig [16], this distinction is not made. In the case of $\kappa=1$, there is a polynomial algorithm by Edmonds [9] to find the optimal pds. In the case of $\kappa>1$, the problem of minimizing the KL divergence is $\mathcal{NP}$–complete. The search algorithm in the BOA and LFDA for that case, which we refer to as graph–search, is greedy and approximate, like the chain–search. Starting from a pds with $\forall_{i\in\mathcal{L}}\langle\pi(i)=\emptyset\rangle$, variables Z_{ω_j} and Z_{ω_k} are iteratively selected for the purpose of enforcing $\omega_k\in\pi(\omega_j)$. This selection is constrained to finding a dag where arcs (v_0,v_1) imply $v_0\in\pi(v_1)$. The selection of ω_k and ω_j each iteration, is such that the sum of conditional entropies from equation 5 is decreased the most and the constraints of equation 3 are not violated.

There are still other special case algorithms, such as the ECGA by Harik [12] that regards only marginal product probability models $\prod_i\hat{P}(\mathcal{S}_i)$, $\bigcup_i\mathcal{S}_i=\mathcal{L}$, $\forall_{i,j}\langle\mathcal{S}_i\cap\mathcal{S}_j=\emptyset\rangle$. Like the LFDA, the ECGA uses minimum discription length as a search metric. This metric has the advantage that the resulting pds will not be overly complex. In the case of the graph–search algorithm using the KL divergence, this can only be influenced by setting κ because the KL divergence merely gives a distance measure from a certain pds to $(\pi^+,\boldsymbol{\omega}^+)$. In this paper, we only regard the four described algorithms in combination with the KL divergence metric. Details on their implementation can be found elsewhere [4, 6]. We close this section by remarking that we can also use the full joint probability distribution $(\pi^+,\boldsymbol{\omega}^+)$ as the pds. However, as noted in section 3, this is only useful in the case of low dimensional problems and a global pdf.

5 Probability density functions

Next to the pds search algorithms from section 4, we require to specify a pdf to use that underlies the IDEA. It follows from sections 2 and 3 that in order to minimize the KL divergence and generate new samples, we require to know the multivariate differential entropy as well as the conditional variant of the pdf. In this section, we specify a few well known probability density functions along with the required information for them to be used within the IDEA framework. In the continuous case, we also give an example of a joint density over two variables.

We assume that the discrete domain contains n_d integers $\{0, 1, \ldots, n_d - 1\}$ and that we have $n+1$ variables $Z_0, Z_1, \ldots, Z_n$ and N samples $z_0^{(S)i}, z_1^{(S)i}, \ldots, z_n^{(S)i}$, $i \in \boldsymbol{N} = \{0, 1, \ldots, N-1\}$. Also, we let $\mathcal{N} = \{1, 2, \ldots, n\}$, $\boldsymbol{n} = (1, 2, \ldots, n)$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and write $e_0 \sqcup (e_1, e_2, \ldots, e_m) = (e_0, e_1, e_2, \ldots, e_m)$.

In the case of discrete data, estimating the probability that a certain variable takes on a certain value can be done by counting frequencies in the sample set:

$$m(\boldsymbol{\nu}, \boldsymbol{\lambda}) = \tag{12}$$
$$\sum_{q=0}^{N-1} \begin{cases} 1 & \text{if } \forall_{i \in \{0,1,\ldots,|\boldsymbol{\lambda}|-1\}} \langle x_{\nu_i}^{(S)q} = \lambda_i \rangle \\ 0 & \text{otherwise} \end{cases}$$

The conditional pdf and the multivariate differential entropy can then be stated as follows[1]:

$$\boldsymbol{p}(x_0 | \{x_i | i \in \mathcal{N}\}) = \frac{m(0 \sqcup \boldsymbol{n}, x_0 \sqcup \boldsymbol{x})}{m(\boldsymbol{n}, \boldsymbol{x})} \tag{13}$$

$$H(\{X_i | i \in \mathcal{N}\}) = \tag{14}$$
$$-\sum_{x_1=0}^{n_d-1} \sum_{x_2=0}^{n_d-1} \cdots \sum_{x_n=0}^{n_d-1} \frac{m(\boldsymbol{n}, \boldsymbol{x})}{N} \ln\left(\frac{m(\boldsymbol{n}, \boldsymbol{x})}{N}\right)$$

The histogram distribution, for which an example is depicted in figure 1, is simply a continuous version of the discrete case. Assume that we have r bins and that $\forall_{i \in \mathcal{N}} \langle \forall_{k \in \boldsymbol{N}} \langle \min^i \leq y_i^{(S)k} < \max^i \rangle \rangle$. Let $\beta^i = (\max^i - \min^i)/r$, $\varphi(x, i) = (x - \min^i)/\beta^i$ and $\boldsymbol{j} = (j_1, j_2, \ldots, j_n)$. We now define a frequency count:

$$b_{\boldsymbol{n}}[\boldsymbol{j}] = \tag{15}$$
$$|\{Y_1^{(S)k} | k \in \boldsymbol{N} \wedge \forall_{q \in \mathcal{N}} \langle j_q \leq \varphi(y_q^{(S)k}, q) \leq j_q + 1 \rangle\}|$$

[1] We write $\boldsymbol{p}(\cdot)$ and $H(\cdot)$ in the discrete case instead of $\boldsymbol{f}(\cdot)$ and $h(\cdot)$.

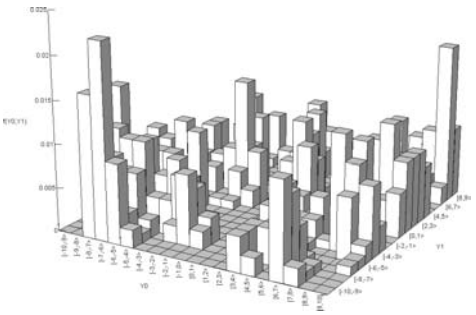

Figure 1: The joint histogram distribution.

The conditional pdf and the multivariate differential entropy can then be stated as follows:

$$\boldsymbol{f}(y_0 | \{y_i | i \in \mathcal{N}\}) = \tag{16}$$
$$\frac{b_{0 \sqcup \boldsymbol{n}}[(\varphi(y_0, 0), \varphi(y_1, 1), \ldots, \varphi(y_n, n))]}{b_{\boldsymbol{n}}[(\varphi(y_1, 1), \varphi(y_2, 2), \ldots, \varphi(y_n, n))]}$$

$$h(\{Y_i | i \in \mathcal{N}\}) = -\frac{\prod_{i=1}^n (\max^i - \min^i)}{r^n} \times \tag{17}$$
$$\sum_{j_1=0}^{r-1} \sum_{j_2=0}^{r-1} \cdots \sum_{j_n=0}^{r-1} \frac{b_{\boldsymbol{n}}[\boldsymbol{j}]}{N} \ln\left(\frac{b_{\boldsymbol{n}}[\boldsymbol{j}]}{N}\right)$$

A widely used parametric continuous pdf is that of the normal distribution, for which an example is depicted in figure 2. Let $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$, $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n) = (E[y_1], E[y_2], \ldots, E[y_n])$ and $\boldsymbol{\Sigma} = E[((y_0 \sqcup \boldsymbol{y}) - (\mu_0 \sqcup \boldsymbol{\mu}))^T ((y_0 \sqcup \boldsymbol{y}) - (\mu_0 \sqcup \boldsymbol{\mu}))]$. By using the notation $\sigma'_{ij} = (\boldsymbol{\Sigma}^{-1})(i, j)$, the conditional pdf and the entropy can be stated as follows:

$$\boldsymbol{f}(y_0 | \{y_i | i \in \mathcal{N}\}) = \frac{1}{\tilde{\sigma}_0 \sqrt{2\pi}} e^{\frac{-(y_0 - \tilde{\mu}_0)^2}{2\tilde{\sigma}_0^2}} \tag{18}$$
$$\text{where } \begin{cases} \tilde{\sigma}_0 & = & \frac{1}{\sqrt{\sigma'_{00}}} \\ \tilde{\mu}_0 & = & \frac{\mu_0 \sigma'_{00} - \sum_{i=1}^n (y_i - \mu_i) \sigma'_{i0}}{\sigma'_{00}} \end{cases}$$

$$h(\{Y_i | i \in \mathcal{N}\}) = \tag{19}$$
$$\frac{1}{2}(n + \ln((2\pi)^n (\det E[(\boldsymbol{y} - \boldsymbol{\mu})^T (\boldsymbol{y} - \boldsymbol{\mu})])))$$

The non–parametric normal kernels pdf, for which an example is depicted in figure 3, places a normal pdf from equation 18 over every available sample point. Let $\mathfrak{s}_i$ be a fixed standard deviation. The conditional pdf and the entropy can then be stated as follows:

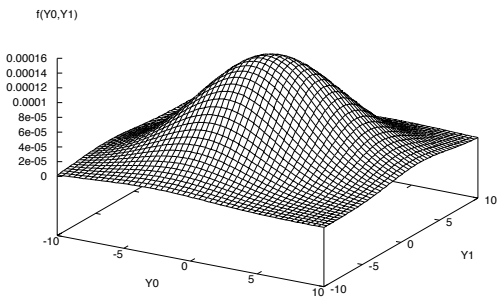

Figure 2: The joint normal distribution.

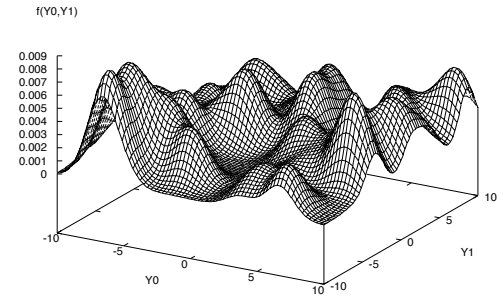

Figure 3: The joint normal kernels distribution.

$$\boldsymbol{f}(y_0|\{y_i|i\in\mathcal{N}\}) = \sum_{i=0}^{N-1}\nu_i\frac{1}{\mathfrak{s}_0\sqrt{2\pi}}e^{\frac{-(y_0-y_0^{(S)i})^2}{2\mathfrak{s}_0^2}} \quad (20)$$

$$\text{where } \nu_i = \frac{e^{-\sum_{j=1}^{n}\frac{(y_j-y_j^{(S)i})^2}{2\mathfrak{s}_j^2}}}{\sum_{k=0}^{N-1}e^{-\sum_{j=1}^{n}\frac{(y_j-y_j^{(S)k})^2}{2\mathfrak{s}_j^2}}}$$

$$h(\{Y_i|i\in\mathcal{N}\}) = \frac{1}{2}\ln\left(N^2(2\pi)^n\prod_{j=0}^{n-1}\mathfrak{s}_j^2\right) - \quad (21)$$

$$\int f(\boldsymbol{y})\ln\left(\sum_{i=0}^{N-1}e^{-\sum_{j=0}^{n-1}\frac{(y_j-y_j^{(S)i})^2}{2\mathfrak{s}_j^2}}\right)d\boldsymbol{y}$$

Each pdf has its own characteristics and complexity. Using histograms, we can arbitrarily well estimate the density of a set of sample points by increasing r. This however comes at the expense of an exponential running time in r^κ and loss of generalization. To this end, the normal pdf is very efficient. Because of its high level of generalization, even when using $(\pi^+, \boldsymbol{\omega}^+)$, the running time is polynomial. For the normal kernels distribution, the running time is still polynomial, but substantially greater than when using the normal pdf.

For the derivation of equations 12, 13, 14, 15, 16, 17, 18 and 20, as well as a more extensive description of the described density functions, see previous work [4, 6]. For the derivation of equations 19 and 21, see for instance [8].

6 Experiments

The continuous function optimization problems we used for testing are the following:

C_0	$\sum_{i=0}^{l-1}(Y_i-1)^2$	$[-5,5]^l$
C_1	$\sum_{i=0}^{l-1}\lfloor Y_i+0.5\rfloor^2$	$[-5,5]^l$
C_2	$\frac{1}{4000}\sum_{i=0}^{l-1}(Y_i-100)^2 -$ $\prod_{i=0}^{l-1}\cos(\frac{Y_i-100}{\sqrt{i+1}})+1$	$[-600,600]^l$
C_3	$-\sum_{i=0}^{l-1}\sin(Y_i)\sin^{20}(\frac{(i+1)Y_i^2}{\pi})$	$[0,\pi]^l$
C_4	$\gamma_i = \frac{24}{1000}(i+2) - Y_i$	$[-3,3]^l$
C_5	$\gamma_0 = Y_0, \gamma_i = Y_i + \gamma_{i-1}$	$[-3,3]^l$
C_6	$\gamma_0 = Y_0, \gamma_i = Y_i + \sin(\gamma_{i-1})$	$[-3,3]^l$

Function C_0 is the sphere model, C_1 is the stepwise version of the sphere model, C_2 is Griewank's function, C_3 is Michalewicz's function and C_4 is a test function by Baluja, as well are C_5 and C_6. The Baluja functions are all of the form $100/(10^{-5}+\sum_{i=0}^{l-1}|\gamma_i|)$ and should be *maximized*. All other functions should be *minimized*.

In all our testing, we used a *monotonic* IDEA. The effectiveness of density estimation depends heavily on the amount of available samples $\lfloor\tau n\rfloor$. We expect a better performance if this amount goes up. Therefore, we fix τ and increase n. To be more precise, we used the rule of thumb by Mühlenbein and Mahnig [16] for FDA and set τ to 0.3. Furthermore, we let n increase from 25 up to 500 in steps of 25 and allowed each algorithm at most $2\cdot 10^6$ function evaluations. If all of the solutions differed by less than $5\cdot 10^{-7}$, termination was enforced also. All results were averaged over 20 runs. The $\mathfrak{s}_i$ standard deviation parameters for the normal kernels distribution were determined as $(\alpha\cdot\text{range}^i)/\lfloor\tau n\rfloor$ where range^i stands for the maximum sample value in the i–th dimension minus the minimum sample value in the i–th dimension. In our experiments, we used $\alpha = 1$. This is the only external parameter for the normal kernels distribution. The normal distribution has no external parameters. The histogram distribution is parameterized by the amount of bins r. We initially ran tests for $r = 2$ and $r = 5$.

We point out that functions C_0 through to C_4 can be optimized by determining a value for each variable separately. For functions C_5 and C_6, this is not the case. We therefore only used the univariate distribution on

C_0, $l = 5$				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
No	**75**	**0.000000**	**1272.8**	**4.7**
Hi_2	100	0.000000	2592.1	2.7
Hi_5	125	0.000000	2575.8	2.4
Ke	500	0.001216	11802.2	3.4

Figure 4: Results on C_0 in 5 dimensions.

C_0, $l = 10$				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
No	**100**	**0.000000**	**2599.20**	**4.6**
Hi_2	125	0.000000	6333.40	2.1
Hi_5	175	0.000000	5679.25	1.9
Ke	500	0.005410	18471.20	3.4

Figure 5: Results on C_0 in 10 dimensions.

C_0 through to C_4 and used other search algorithms as well on C_5 and C_6.

We tackled C_0 and C_1 in 5 and 10 dimensions using the normal distribution, the histogram distribution with 2 and 5 bins and the normal kernels distribution. Tables 4, 5, 6 and 7 give an overview of the results. The tables show the results for the smallest value of n for which the average best cost was equal to the optimal value within the demanded precision. If this value was not found, the results for $n = 500$ are given. The table shows the minimal value for n, the average cost $\overline{C}$, the average amount of function evaluations $\overline{\text{evals}}$ and the relative time RT. Let $\text{FT}(x)$ be the time to perform x function evaluations and $\text{TT}(x)$ the total time spent while those x function evaluations were performed. Then, $\text{RT}(x) = (\text{TT}(x) - \text{FT}(x))/\text{FT}(x)$. We determined RT as $\text{RT}(10^6)$. The best results are printed in boldface.

Functions C_0 and C_1 are quite simple. It seems that for very smooth functions at least, the normal distribution works well. The normal kernels distribution seems too much cluster oriented to work well on these functions. However, by increasing α, its effectiveness on smooth functions will be increased. Empirical verification has shown that for $\alpha \approx 1.7$, about 3400 evaluations are

C_1, $l = 5$				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
No	75	0.000000	697.75	2.3
Hi_2	50	0.000000	327.20	1.3
Hi_5	**50**	**0.000000**	**303.80**	**1.3**
Ke	75	0.000000	864.70	2.2

Figure 6: Results on C_1 in 5 dimensions.

C_1, $l = 10$				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
No	**50**	**0.000000**	**658.40**	**2.2**
Hi_2	75	0.000000	822.30	0.9
Hi_5	75	0.000000	697.75	0.9
Ke	125	0.000000	2514.20	1.7

Figure 7: Results on C_1 in 10 dimensions.

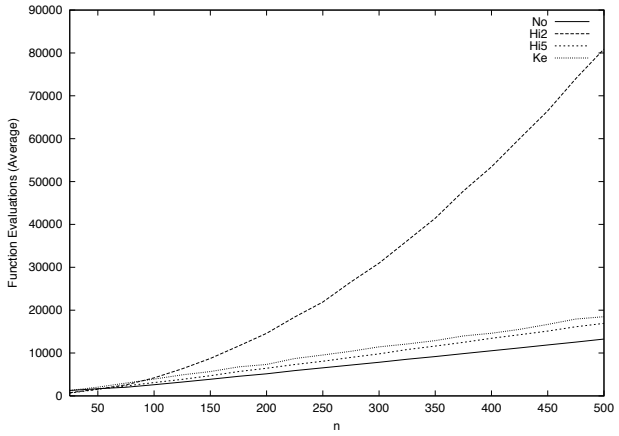

Figure 8: $\overline{\text{eval}}$ for C_0 in 10 dimensions.

required to solve C_0 with $l = 10$ to minimality.

The histogram distribution also seems to work well. However, even though Hi_2 requires at most the same population size as Hi_5, the amount of required evaluations is larger. Judging by empirical results, we need a larger amount of bins than merely two (or one) in order for the amount function evaluations to scale up. This can be seen in figures 8 and 9, which show the average amount of evaluations.

We continue our testing with only $r = 5$ in the case of the histogram distribution. Function C_2 is less smooth than the previous two, but not as rugged as C_3. For $l = 5$, function C_2 gave some problems for all approaches and for $l = 10$ for the normal kernels and the histogram approaches. The problem was that they required a very large amount of function evaluations. When the maximum of function evaluations is reached, increasing n will give worse results. This is because the search becomes slower and termination occurs before convergence. Therefore, tables 10 and 11 show the best average result, instead of the result for $n = 500$, if the maximum of evaluations dominates termination and the minimum of 0 has not been reached within the demanded precision.

The normal kernels distribution does not give the best approximate solution, which might be because of the choice of α. Empirical verification has shown that for

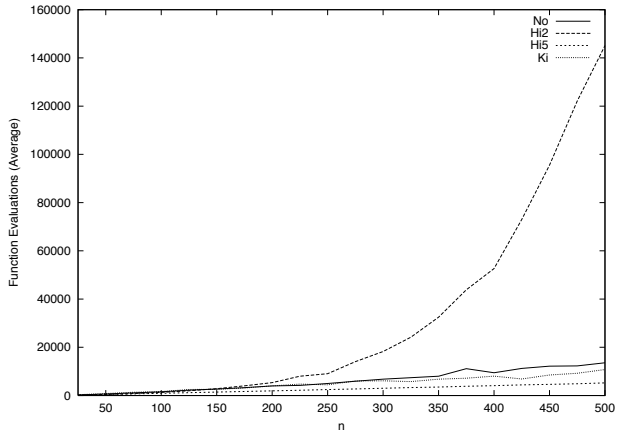

Figure 9: $\overline{\text{eval}}$ for C_1 in 10 dimensions.

C_2, $l = 5$				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
No	150	0.000639	1003228.00	1.2
Hi_5	**400**	**0.000000**	**849919.20**	**0.8**
Ke	300	0.015769	197257.95	0.9

Figure 10: Results on C_2 in 5 dimensions.

$\alpha \approx 3$, about 30000 evaluations are required to solve C_2 with $l = 10$ to minimality. The amount of evaluations on functions C_0 and C_1 were observed to be just about the same as the other approaches, with the exception of Hi_2. In the case of C_2, the amount of function evaluations is the least for the normal kernels distribution. The amount of evaluations needed by the normal distribution becomes a lot larger when the global smoothness of the optimization function is decreased. Figure 12 shows the least amount of function evaluations for $l = 10$ on function C_2, which is more interesting than the average, because the average is mostly dominated by the maximum we set.

The advantage of the normal kernels distribution becomes more evident when we look at the results on C_3, which is more epistatic than C_2. An overview is given in tables 13 and 14. For $l = 5$, the minimum found by all methods is -4.687658 and for $l = 10$ the minimum found by the normal kernels approach is -9.660152. Because none of the methods were able to reach an

C_2, $l = 10$				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
No	**275**	**0.000000**	**62835.95**	**0.9**
Hi_5	450	0.001725	161894.40	0.5
Ke	200	0.178204	99182.00	0.8

Figure 11: Results on C_2 in 10 dimensions.

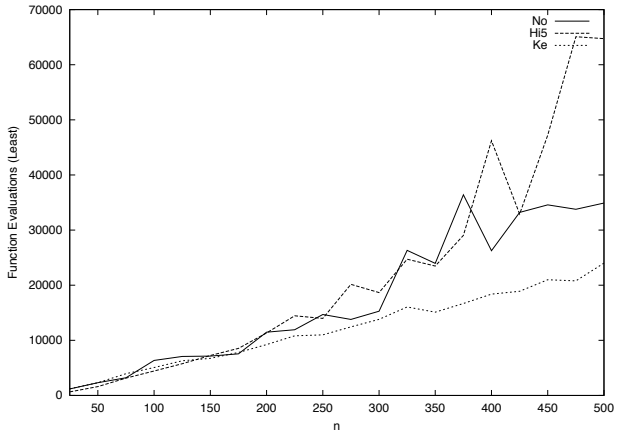

Figure 12: Least amount of function evaluations for C_2 in 10 dimensions.

C_3, $l = 5$				
pdf	n	$\overline{C}$	$\overline{\text{evals}}$	RT
No	500	-4.646790	140882.45	0.6
Hi_5	500	-4.668619	24087.20	0.5
Ke	**500**	**-4.687645**	**10696.55**	**0.5**

Figure 13: Results on C_3 in 5 dimensions.

average of these minima within the demanded precision, the results are given for $n = 500$. The normal distribution suffered again from the problem that the maximum amount of function evaluations was already reached for small values of n. In figure 15 the least amount of function evaluations required over 20 runs is shown for the 10 dimensional case. The advantage of the normal kernels approach in more epistatic search spaces is clear.

Note that the histogram distribution also requires less function evaluations. It is clear that the histogram distribution can allow for more detail by increasing r, which has somewhat of the same role as α in the case of the normal kernels distribution. However, if we move to functions that have interacting variables, allowing for this in a pds results in a running time of $\mathcal{O}(r^\kappa)$ for the histogram distribution [4], which doesn't scale up as well as the other distributions. We therefore disregard the histogram distribution from now on.

C_3, $l = 10$				
pdf	n	$\overline{C}$	$\overline{\text{evals}}$	RT
No	500	-9.428565	1346040.95	0.6
Hi_5	500	-9.533540	708870.65	0.5
Ke	**500**	**-9.659621**	**18120.20**	**0.5**

Figure 14: Results on C_3 in 10 dimensions.

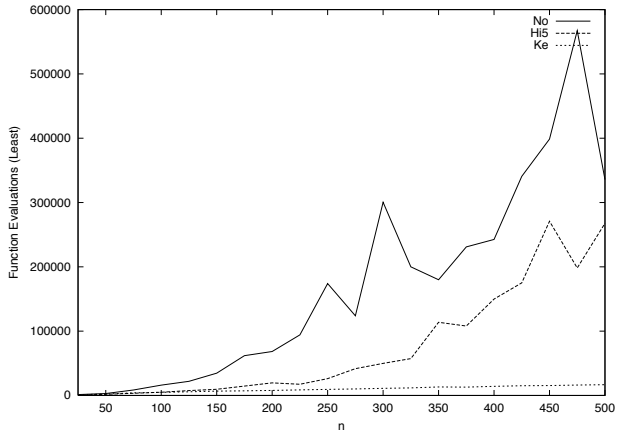

Figure 15: Least amount of function evaluations for C_3 in 10 dimensions.

C_2, $l = 100$			
Method	$\overline{C_2}$	n	RT
$(10+50)$–ES	399.07	—	—
PBIL (Binary)	16.43	—	—
PBIL (Gray)	366,77	—	—
PBIL_C	4803	—	—
IDEA No_U	9999999.865194	225	5.85
IDEA No_C	9999999.902504	250	20.81
IDEA No_G	**9999999.962383**	**350**	**150.67**
IDEA No_FC	9999999.876634	350	6.72

Figure 16: Results on C_4 in 100 dimensions.

On functions C_4, C_5 and C_6, we applied IDEAs with the univariate distribution, the chain–search algorithm, the exact graph–search algorithm for $\kappa = 1$ and a fixed chain that links the variables in the same way as the functions are defined, so $\omega_i = l - i - 1$, $\pi(0) = \emptyset$ and $\pi(i) = i - 1$, $i \geq 1$. This latter pds is the best possible pds as it represents exactly the structure of the problem, which gives an upper bound. The dimensionality of the problem is set to $l = 100$ and the maximum amount of function evaluations is 200000. Repeating earlier reported results [22], tables 16, 17 and 18 show that our approaches obtain better results. In the table, No_U uses the univariate distribution, No_C uses chain–search, No_G uses graph–search with $\kappa = 1$ and No_FC uses the fixed chain, all of which use the normal pdf.

7 Discussion

The use of the normal kernels pdf is clearly dependent on the value of α. It is intuitively clear that a larger α results in an algorithm that is better applied to problems that are globally more smooth. Adapting the value of α to the level of epistasis of the fitness landscape during a run might lead to very efficient EAs.

C_3, $l = 100$			
Method	$\overline{C_3}$	n	RT
$(10+50)$–ES	2.91	—	—
PBIL (Binary)	2.12	—	—
PBIL (Gray)	2.62	—	—
PBIL_C	4.76	—	—
IDEA No_U	4.513876	150	6.57
IDEA No_C	5.296601	200	23.83
IDEA No_G	**7.498911**	**275**	**56.75**
IDEA No_FC	13.483005	350	7.82

Figure 17: Results on C_5 in 100 dimensions.

C_4, $l = 100$			
Method	$\overline{C_4}$	n	RT
$(10+50)$–ES	7.56	—	—
PBIL (Binary)	4.4	—	—
PBIL (Gray)	5.61	—	—
PBIL_C	11.18	—	—
IDEA No_U	13.398891	250	0.96
IDEA No_C	14.851334	300	3.75
IDEA No_G	**27.730714**	**550**	**5.00**
IDEA No_FC	49.651935	450	1.08

Figure 18: Results on C_6 in 100 dimensions.

One fundamental flaw of the normal kernels pdf is known to be its tendency to overfit a distribution. This can be somewhat regulated by α, but not entirely. Samples drawn from a single normal distribution can result in a normal kernels estimation far from the smooth original. This is one of the reasons why a normal mixture distribution based on M normal kernels will most likely provide very useful properties and prevent overfitting.

IDEAs take up more time as the allowed complexity of the pds goes up. It is therefore important to be aware of the running times of the parts of an algorithm on top of the amount of function evaluations. To this end, we have already used the notion of *relative time* RT. Moreover, we note that all of the proposed search algorithms as well as the estimation and sampling algorithms are polynomial in all parameters for the normal distribution as well as the normal kernels distribution. This is *not* the case for the histogram distribution and the mentioned distribution in the discrete case, as they are exponential in $\mathcal{O}(r^\kappa)$ where in the discrete case we have $r = 2$. Given that we require quite a few bins to efficiently optimize a problem using the histogram distribution, methods based upon that distribution do not scale up for higher order epistatic problems.

8 Conclusions

The IDEA framework allows for elegant modelling of algorithms that perform evolutionary optimization based on density estimation. We have shown how this can be done for three different probability density functions in the continuous case as well as for one pdf in the discrete case. The experiments indicate that building and using probabilistic models in the case of continuous optimization problems can be effective. By using a pds in which variables are allowed to interact, IDEAs can be constructed that perform very well on continuous and epistatic problems.

References

[1] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russell, editors, *Proceedings of the twelfth International Conference on Machine Learning*, pages 38–46. Morgan Kauffman publishers, 1995

[2] S. Baluja and S. Davies. Using optimal dependency–trees for combinatorial optimization: Learning the structure of the search space. In D.H. Fisher, editor, *Proceedings of the 1997 International Conf. on Machine Learning*. Morgan Kauffman publishers, 1997

[3] J.S. De Bonet, C. Isbell, and P. Viola. Mimic: Finding optima by estimating probability densities. *Advances in Neural Information Processing*, 9, 1996

[4] 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

[5] 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, *Proceedings of the GECCO–1999 Genetic and Evolutionary Computation Conference*, pages 60–5.97. Morgan Kaufmann Publishers, 1999

[6] 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

[7] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Inf. Theory*, 14:462–467, 1968

[8] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons Inc., 1991

[9] J. Edmonds. Optimum branchings. *J. Res. Nat. Bur. Standards*, 71B:233–240, 1967. Reprinted in Math. of the Decision Sciences, *Amer. Math. Soc. Lectures in Appl. Math.*, 11:335–345, 1968

[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, *Proceedings of the GECCO–1999 Genetic and Evolutionary Computation Conference*, pages 840–846. Morgan Kaufmann Publishers, 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. Rep. 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 *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pages 523–528. IEEE Press, 1998

[14] J.H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor: Univ. of Michigan Press, 1975

[15] S. Kullback. *Information Theory And Statistics*. New York: Dover, 1968

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

[17] H. Mühlenbein, T. Mahnig, and O. Rodriguez. Schemata, distributions and graphical models in evolutionary optimization. *J. of Heur.*, 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, *Proceedings of the GECCO–1999 Genetic and Evolutionary Computation Conf.*, pages 525–532. Morgan Kaufmann, 1999

[20] M. Pelikan, D.E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. IlliGAL Tech. Rep. 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, *Proceedings of Artificial Evolution '97*, pages 137–144. Springer Verlag, LNCS 1363, 1997

[24] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–5.956, 1948