Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 361 (2006) 57–71
Theoretical Computer Science
www.elsevier.com/locate/tcs

Compact representations as a search strategy: Compression EDAs

Marc Toussaint*

Institute for Adaptive and Neural Computation, University of Edinburgh, 5 Forrest Hill, Edinburgh EH1 2QL, Scotland, UK

Abstract

The choice of representation crucially determines the capability of search processes to find complex solutions in which many variables interact. The question is how good representations can be found and how they can be adapted online to account for what can be learned about the structure of the problem from previous samples. We address these questions in a scenario that we term indirect Estimation-of-Distribution: We consider a decorrelated search distribution (mutational variability) on a variable length genotype space. A one-to-one encoding onto the phenotype space then needs to induce an adapted phenotypic search distribution incorporating the dependencies between phenotypic variables that have been observed successful previously. Formalizing this in the framework of Estimation-of-Distribution Algorithms, an adapted phenotypic search distribution can be characterized as minimizing the Kullback–Leibler divergence (KLD) to a population of previously selected samples (parents). The paper derives a relation between this KLD and the description length of the encoding, stating that compact representations provide a way to minimize the divergence. A proposed class of Compression Evolutionary Algorithms and experiments with an grammar-based compression scheme illustrate the new concept.

Keywords: Estimation-of-Distribution Algorithms; Factorial representations; Compression; Minimal description length; Evolutionary Algorithms; Genotype–phenotype mapping

1. Introduction

The complexity of a problem largely depends on the interactions between the variables of a solution. A stochastic search process will perform well on a complex problem only when the search distribution is adapted to these interactions, i.e., when the search distribution obeys these dependencies between variables.

One approach to design a search distribution is to choose a suitable representation. The questions are how to find representations that induce the desired dependencies on the search space, and how they can be adapted online to account for what can be learned about the structure of the problem from previous samples. There have been various approaches to characterize what a good representation is, considering, for example, all possible representations, Gray vs. binary codes, redundant representations, and recursive encodings [16,30,4,22,12]. Theoretical approaches concerning the adaptation of the search distribution based on the specific current population include, for example, Estimation-of-Distribution

* Tel.: +44 131 650 3089; fax: +44 131 650 6899.
E-mail address: mtoussai@inf.ed.ac.uk

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.04.005

Algorithms [21], Walsh analysis [11], and Maximal Entropy principles [31]. Also using Minimum Description Length principles for building search distributions has been discussed before [9]. [1]

Our approach is to consider a specific scenario that we term indirect Estimation-of-Distribution: We assume that the search distribution on a variable length coding space (genotype space) is decorrelated. A one-to-one encoding onto the search space (phenotype space) then needs to induce a properly structured phenotypic search distribution. The idea of this scenario is that the encoding receives all responsibilities to induce the structural properties of the phenotypic search distribution, leaving a simple problem of unstructured (decorrelated) adaptation on the genotype level.

We formalize the scenario within the framework of Estimation-of-Distribution Algorithms, where an adapted phenotypic search distribution can be characterized as minimizing the Kullback–Leibler divergence (KLD) to a population of previously selected samples (parents). Our core result is a relation between the KLD and the description length of the encoding in the specific scenario of indirect Estimation-of-Distribution based on a variable length genotype space. The result states that compact representations provide a way to minimize this divergence—and may thus be seen as transferring similar results on Minimum Description Length in the context of modeling, in particular model selection, (e.g., [5,28]) to the specific domain of stochastic search based on adaptive representations.

An intuitive way to grasp these results might be the following: Consider a set of good (selected) samples in phenotype space. In this selected population there will generally exist dependencies between phenotypic variables, measurable as mutual information between them. This is the information that should be extracted and exploited for further search. Assume we can map these samples on variable size strings such that the average string length is minimized. Before the compression there was mutual information between the phenotypic variables. After the compression, there should be no mutual information between the (genetic) symbols that describe the samples because otherwise the mapping would not be a minimum description length compression. Thus, a compression is one way (among others) to map on a representation in which symbols are decoupled (i.e., to map on a factorial representation). A compression can also be considered as an implicit analysis of the dependencies that have been present in the parent population because it is able to dissolve them by introducing new symbols. Eventually, the key idea is that *inverting* the compression is a mechanism to induce exactly these dependencies. For instance, when there is noise (decorrelated mutational variability) on the genetic symbols, this should translate to a phenotypic variability that obeys these dependencies.

The following two sections will introduce the theoretical framework, including the basics of Estimation-of-Distribution Algorithms and the indirect induction of a search distribution. Section 4 derives the main results on the relation between compact representations and Estimation-of-Distribution. Section 5 aims to illustrate the implications by proposing a class of Compression Evolutionary Algorithms and presenting experiments with a Compression EDA and Genetic Algorithm (GA) based on a simple grammar-based compression scheme (well comparable to the ideas of [17]).

2. Estimation-of-Distribution

Let P be the search space. A heuristic search scheme is a stochastic process in the space Λ of distributions over the search space in which a search distribution $q \in \Lambda$ is propagated iteratively. In each iteration, samples from q are drawn, evaluated, and the outcome of evaluation is used to design, according to some heuristic, a new search distribution in the next step.

For instance, in ordinary Evolutionary Algorithms (EAs) the search distribution is given by a finite parent population and recombination and mutation operators (leading to a mixture of mutation kernels as search distribution). The search distribution is sampled, leading to the finite offspring population, which is then evaluated, leading to the selection probability distribution over these offspring. The heuristic to generate the new search distribution in simple EAs is to sample the selection probability distribution, leading to a new parent population which in turn induces a new search distribution. We do not need to specify these operators here explicitly. We develop the theory on the abstract level of search distributions.

What is a reasonable heuristic to design a search distribution given the results of evaluation of previous samples? We will follow here the idea of Estimation-of-Distribution Algorithms [21] which can be described as follows.

---

[1] In the ECGA proposed by [9] MDL refers to finding a partitioning of the phenotypic variables that minimizes the sum of entropies in each partition, which is trivially equivalent to maximizing the sum of mutual informations in each partition (i.e., finding a high linkage partitioning). Finding or adapting a concrete mapping to another coding space is not considered.

We assume that the outcome of evaluation is given as a distribution p over P, which might represent the parent population.[2] Given that the space of feasible search distributions is limited to Q, a simple heuristic to choose the new search distribution is to pick the one that is most similar to p. Similarity can be measured by the KLD $D(p:q) = E_p\{\log p(x)/q(x)\}$ between two distributions, which also captures the structural similarity between two distributions in the sense of the similarity of different order dependencies (see [1], please note the relations between the KLD, log-likelihood, free energy, and mean energy).[3] Thus, one heuristic to design the new search distribution q' reads

$$q' = \operatorname*{argmin}_{q\in Q} D(p:q). \tag{1}$$

We term this specific kind of an EDA *KL-search*. In general, the crucial parameter of KL-search is the choice of the set Q of feasible search distributions. On the one hand, the choice of Q determines the computational cost of the minimization (1) in every step. On the other hand, it determines the algorithm's capability to express and exploit the structure observable in p.

Some algorithms of the class of EDAs are exact instantiations of KL-search: MIMIC [7] chooses Q to be the set of Markov chains, PBIL [2] chooses Q as the set of factorized distributions. Other EDAs differ from KL-search in the choice of the similarity measure (they use alternatives to the KLD, e.g., BOA [20] takes a Bayesian Dirichlet Metric). But all of them can distinctly be characterized by their choice of Q, which may also be the set of dependency trees (COMIT, [3]), Bayesian networks (BOA, [20]), or Bayesian networks with local structures (hBOA, [19]).

Despite its conceptual simplicity, the minimization required in each iteration of KL-search can be computationally very expensive, depending on the complexity of the distributions in Q. When only simple distributions, like factorized distributions (PBIL) or Markov chains (MIMIC) are allowed, the minimization can be calculated directly. For more complex distributions (like Bayesian networks, BOA), the minimization itself requires an iterative procedure.

Finally note that distributions in Q should typically be constrained to have a minimum entropy. In that way, a repeated cycle of entropy decrease (in the course of evaluation) and entropy increase (when picking a new search distribution) ensures exploration and prevents the algorithms from early convergence.

3. Indirect induction of search distributions

In this paper, we are interested in indirect codings of search points. In the heuristic search framework, this means that a distribution $\tilde{q} \in \tilde{\Lambda}$ over a coding space G, the *genotype space*, is maintained. The search distribution q over the actual search space P (*phenotype space*) is then given indirectly via a *coding* $\phi: G \to P$,

$$q(x) = \sum_{g\in[x]_\phi} \tilde{q}(g) \quad \text{where } x \in P,\ [x]_\phi = \{g \in G \mid \phi(g) = x\}. \tag{2}$$

We also use the short notation $q = \tilde{q} \circ \phi^{-1}$ for this projection of $\tilde{q}$ under ϕ. The set $[x]_\phi$ of all genotypes mapping to the same phenotype x is an equivalence class under ϕ, also called *neutral set* of x.

Three additional constraints define the "indirect encoding case" considered in this paper: *First*, we impose that $\phi: G \to P$ shall be bijective (one-to-one). We denote the space of all bijective codings $G \to P$ by Φ. The discussion in Section 6 will establish a relation to non-bijective codings.

Second, G is the space of variable length strings over some finite alphabet $\mathcal{A}$,

$$G = \bigcup_{l=1}^{\infty} \mathcal{A}^l. \tag{3}$$

---

[2] Generally, in this formalism p is meant to encode any information that we receive from evaluations. Typically, p is non-vanishing only on a finite set of samples (the offspring population) and the values of p might be (in a normalized way) the fitness values of these offspring. Alternatively, p might represent a resampling of such a selection distribution, which corresponds to the parent population.

[3] With the definitions of the entropy $H(p) = -E_p\{\log p(x)\}$ and the log-likelihood $\mathcal{L}(q) = E_p\{\log q(x)\}$ we have $D(p:q) = -H(p) - \mathcal{L}(q)$. One could roughly say, "minimizing the KLD means maximizing the log-likelihood *and* the entropy". Further, when defining an energy functional $E(x) = -\log q(x)$, the mean energy $E = E_p\{E(x)\} = -\mathcal{L}(q)$ is the negative log-likelihood while the free energy $F = E - H(p)$ is the KLD.

Table 1
Basic notations

In P	In G	Description
x	$g = (g_1, \dots, g_l)$	A sample (a string in the case of G)
Λ	$\tilde{\Lambda}$	Space of distributions
Q	$\tilde{Q}$	Space of feasible search distributions
p	$\tilde{p} = p \circ \phi$	Distribution to be estimated (selected population)
q	$\tilde{q} = q \circ \phi$	Search distribution
	$\tilde{q}(g) = \tilde{q}(l) \prod_i \tilde{q}_i^l(g_i)$	Factorization of feasible search distribution on G

It is technically unclear how to define a marginal over the ith symbol (or mutual information between symbols) directly for a variable length distribution $\tilde{q} \in \tilde{\Lambda}$. Thus we will consider the decomposition

$$\tilde{q}(g) = \tilde{q}(g|l)\,\tilde{q}(l), \quad \text{where } g \in G,\ l = \text{length}(g) \in \mathbb{N}. \tag{4}$$

Here, $\tilde{q}(l)$ is a distribution over the genotype length $l \in \mathbb{N}$, and $\tilde{q}(g|l)$ the conditional distribution over a fixed length alphabet $\mathcal{A}^l$. We use the short notation $\tilde{q}^l \equiv \tilde{q}(\cdot|l)$ for this length-conditioned distribution. The marginal $\tilde{q}_i^l$ over the ith symbol ($i \leqslant l$) and the mutual information $I(\tilde{q}^l) = \sum_i H(\tilde{q}_i^l) - H(\tilde{q}^l)$ can then be defined as usual in terms of the marginal entropies and total entropy. [4]

As the *third* constraint we limit the space of possible search distributions in a certain way: We impose that the length-conditioned distributions $\tilde{q}^l$ on the genotype space have to factorize, i.e.,

$$\tilde{q}^l(g) = \prod_{i=1}^{l} \tilde{q}_i^l(g_i), \tag{5}$$

or, equivalently, that the mutual information $I(\tilde{q}^l)$ vanishes. We denote the set of feasible genotype distributions by

$$\tilde{Q} \subseteq \{\tilde{q} \in \tilde{\Lambda} \mid \forall l :\ I(\tilde{q}^l) = 0\}. \tag{6}$$

The space $Q \subseteq \Lambda$ of feasible search distribution over P then is

$$Q = \{\tilde{q} \circ \phi^{-1} \mid \phi \in \Phi,\ \tilde{q} \in \tilde{Q}\}. \tag{7}$$

Table 1 recollects the basic notation we have introduced. In summary, indirect induction of the search distribution means that, in order to design a search distribution $q \in \Lambda$ we have to pick a bijective coding $\phi \in \Phi$ and a decorrelated distribution $\tilde{q} \in \tilde{Q}$ on the genotype space. In other terms, every feasible search distribution q corresponds to a pair $(\phi, \tilde{q})$.

4. Indirect Estimation-of-Distribution via compression

KL-search proposes how to pick a new search distribution out of Q incorporating the knowledge on the evaluation of previous samples. In the previous section we specified a Q that defines the indirect coding case, where a choice of q means to pick a bijective coding ϕ and a factorized distribution $\tilde{q}$ on G. Combining this, KL-search amounts to a heuristic to pick a coding ϕ (and a $\tilde{q}$) such that knowledge on previous evaluations is incorporated. In this section we will derive results on how this heuristic to pick a coding reads more explicitly. We first discuss the simpler fixed length case before addressing the general one:

Fixed length case: Assume that G contains only strings of fixed length l, $G = \mathcal{A}^l$. Then the marginals $\tilde{q}_i$ are straight-forward to define and $I(\tilde{q}) = \sum_i H(\tilde{q}_i) - H(\tilde{q}) = 0$ constrains $\tilde{q}$ to vanishing dependencies between genes.

[4] For a distribution q over some product space $\mathcal{A} \times \cdots \times \mathcal{A}$, we generally denote the marginal over the ith variable by q_i. The mutual information $I(q) = \sum_i H(q_i) - H(q)$ measures all dependencies between variables of any order (not only pair-wise dependencies).

Given Definition (7) of Q for the case of indirect codings, we have

$$D(p:q) = \sum_x p(x)\ \ln \frac{p(x)}{\sum_{g' \in [x]_\phi} \tilde{q}(g')} = \sum_g \tilde{p}(g)\ \ln \frac{\tilde{p}(g)}{\tilde{q}(g)}. \tag{8}$$

Here, we defined $\tilde{p}$ as the back-projection of p onto the coding space G, $\tilde{p} = p \circ \phi$. The last step uses that ϕ is bijective such that there exists exactly one $g \in [x]_\phi$. Next we expand Eq. (8) into two terms and use that $\tilde{q}$ has to factorize (5),

$$D(p:q) = \sum_g \tilde{p}(g)\ \ln \frac{\tilde{p}(g)}{\prod_i \tilde{p}_i(g_i)} + \sum_g \tilde{p}(g) \ln \frac{\prod_i \tilde{p}_i(g_i)}{\prod_i \tilde{q}_i(g_i)} = I(\tilde{p}) + D(\tilde{p}_{(1)} : \tilde{q}). \tag{9}$$

Here we defined $\tilde{p}_{(1)}(g) = \prod_i\ \tilde{p}_i(g_i)$ as the "factorized reduction" of $\tilde{p}$ (see also [1] on how to generally define the kth order reduction $\tilde{p}_{(k)}$ of $\tilde{p}$ containing all and only the dependencies of order $\leqslant k$).

This result states that, in order to follow the KL-search scheme (1) in the indirect coding case, one should find a coding ϕ and a search distribution $\tilde{q}$ such that $I(\tilde{p}) + D(\tilde{p}_{(1)} : \tilde{q})$ is minimized.

Here, I would like to distinguish two cases. In the first case we assume that $\tilde{Q}$ comprises *all* factorized distributions without a bound on the entropy (equality in Eq. (6)). In this case, no matter which ϕ is chosen, one can always minimize $D(\tilde{p}_{(1)} : \tilde{q})$ to zero by picking $\tilde{q} = \tilde{p}_{(1)}$. Since $I(\tilde{p})$ is independent of $\tilde{q}$, the minimization (1) can be realized by first optimizing ϕ and then picking $\tilde{q}$:

$$\operatorname*{argmin}_{q \in Q} D(p:q) = (\phi, \tilde{q}), \quad \text{where } \phi = \operatorname*{argmin}_{\phi} I(\tilde{p}) \text{ and } \tilde{q} = \tilde{p}_{(1)}. \tag{10}$$

We call this procedure (first optimizing ϕ, then picking $\tilde{q}$) the *two step procedure*. Note that $\tilde{p}_{(1)}$ in the last equation depends on the ϕ chosen before.

However, in a realistic algorithm, $\tilde{Q}$ should not comprise all factorized distributions but obey a lower bound on the entropy of these distributions to ensure exploration.[5] Hence, in the second case, when $\tilde{Q}$ is only a subset of all factorized distributions, $D(\tilde{p}_{(1)} : \tilde{q})$ can generally not be minimized to zero and the minimization of (9) can *not exactly* be decomposed in the two steps of first minimizing $I(\tilde{p})$ w.r.t. ϕ, and then, for a fixed ϕ, minimizing $D(\tilde{p}_{(1)} : \tilde{q})$ w.r.t. $\tilde{q}$. The exact minimization of (9) remains a coupled problem of finding a pair $(\phi, \tilde{q})$.

For completeness, let us estimate a bound on the "error" made when still adopting the two step procedure of minimization. Let $(\phi^*, \tilde{q}^*)$ be a coding and genotype distribution that indeed minimize (9); and let $(\phi', \tilde{q}')$ be the result of the two step procedure, i.e., ϕ' minimizes $I(\tilde{p}')$ and $\tilde{q}'$ minimizes $D(\tilde{p}'_{(1)} : \tilde{q}')$ for the given coding ϕ'. Here, $\tilde{p}' = p \circ \phi'$ and $\tilde{p}^* = p \circ \phi^*$. A rough bound for the error made can be estimated as follows, to be explained in detail below

$$\begin{aligned} D(p:q') - D(p:q^*) &= D(\tilde{p}'_{(1)} : \tilde{q}') - D(\tilde{p}^*_{(1)} : \tilde{q}^*) + I(\tilde{p}') - I(\tilde{p}^*) \\ &\leqslant D(\tilde{p}'_{(1)} : \tilde{q}') - D(\tilde{p}^*_{(1)} : \tilde{q}^*) \leqslant D(\tilde{p}'_{(1)} : \tilde{q}') \\ &= \sum_{i=1}^{l} D(\tilde{p}'_i : \tilde{q}'_i) \leqslant -\sum_{i=1}^{l} \log \tilde{q}'_i(a_i)\ =\ -l \log(1-\alpha) \leqslant l\,\hbar. \end{aligned} \tag{11}$$

The first inequality stems from the fact that ϕ' minimizes $I(\tilde{p}')$ and thus $I(\tilde{p}') \leqslant I(\tilde{p}^*)$. Since both, $\tilde{p}'_{(1)}$ and $\tilde{q}'$, are factorized distributions, their KLD decomposes into a sum. For each marginal, when there is a lower bound $\hbar$ on the entropy of $\tilde{q}'_i$, the divergence $D(\tilde{p}'_i : \tilde{q}'_i)$ is particularly large when $\tilde{p}'_i$ has very low entropy. In the worst case, $\tilde{p}'_i$ has zero entropy, i.e., is non-zero only for a single symbol $a_i \in \mathcal{A}$. In that case $D(\tilde{p}'_i : \tilde{q}'_i) = -\log \tilde{q}'_i(a_i)$. In order to minimize $D(\tilde{p}'_i : \tilde{q}'_i)$, $\tilde{q}'_i$ is chosen to have the form of the typical symbol mutation distribution with mutation rate α, where $\tilde{q}'_i(a_i) = 1-\alpha$ and $\tilde{q}'_i(a) = \alpha/|\mathcal{A}| - 1$ for $a \neq a_i$. Then, $H(\tilde{q}_i) = -(1-\alpha)\ \log(1-\alpha) - \alpha \log \alpha/|\mathcal{A}| - 1$.

[5] Recall that for $\tilde{q} = \tilde{p}_{(1)}$, and since ϕ is bijective, $H(p) = H(\tilde{p}) = H(\tilde{p}_{(1)}) - I(\tilde{p}) = H(\tilde{q}) - I(\tilde{p}) = H(q) - I(\tilde{p})$. Therefore, the entropy of search $H(q)$ would only be greater than $H(p)$ by the amount of $I(\tilde{p})$, which is minimized.

Given the lower bound $\hbar$ on the entropy, the minimal mutation rate α can be chosen to ensure $H(\tilde{q}_i) = \hbar$ and $D(\tilde{p}_i : \tilde{q}_i) = -\log(1-\alpha) \leqslant \hbar$. Thus, in the worst case, the "error" made when using the two step procedure instead of the exact minimization of (9) is at most $l\,\hbar$.

Variable length case: Let us repeat the above derivations in the general case when $G = \bigcup_{l=1}^{\infty} \mathcal{A}^l$ comprises strings of any length over the alphabet $\mathcal{A}$. The constraint of vanishing mutual information in $\tilde{q}$ now refers to the length-conditioned distributions $\tilde{q}^l$, i.e., we impose $I(\tilde{q}^l) = 0$ while $\tilde{q}(l)$ is unconstrained. (Recall $\tilde{q}(g) = \tilde{q}^l(g)\,\tilde{q}(l)$ where $l = \text{length}(g)$.) Eq. (8) now leads to

$$\begin{aligned} D(p:q) &= \sum_l \sum_g \tilde{p}^l(g)\tilde{p}(l)\,\ln\frac{\tilde{p}^l(g)\tilde{p}(l)}{\tilde{q}^l(g)\tilde{q}(l)} \\ &= \sum_l \tilde{p}(l) \sum_g \tilde{p}^l(g)\,\ln\frac{\tilde{p}^l(g)}{\tilde{q}^l(g)} + \sum_l \tilde{p}(l)\,\ln\frac{\tilde{p}(l)}{\tilde{q}(l)} \\ &= \sum_l \tilde{p}(l) \sum_g \tilde{p}^l(g)\,\ln\frac{\tilde{p}^l(g)}{\tilde{p}^l_{(1)}(g)} + \sum_l \tilde{p}(l) \sum_g \tilde{p}^l(g)\,\ln\frac{\tilde{p}^l_{(1)}(g)}{\tilde{q}^l(g)} + D(\tilde{p}(l):\tilde{q}(l)) \\ &= E_l\left\{I(\tilde{p}^l)\right\} + E_l\left\{D(\tilde{p}^l_{(1)} : \tilde{q}^l)\right\} + D(\tilde{p}(l):\tilde{q}(l)), \end{aligned} \tag{12}$$

where we introduced $E_l\{\cdot\}$ as the expectation w.r.t. $\tilde{p}(l)$ (which depends on ϕ). The entropy of p can be written as

$$\begin{aligned} H(p) &= -\sum_g p(g)\ln p(g) = -\sum_l \tilde{p}(l) \sum_g \tilde{p}^l(g)\ln[\tilde{p}^l(g)\,\tilde{p}(l)] \\ &= -\sum_l \tilde{p}(l)\left[\sum_g \tilde{p}^l(g)\ln\tilde{p}^l(g) + \sum_g \tilde{p}^l(g)\ln\tilde{p}(l)\right] \\ &= \sum_l \tilde{p}(l) H(\tilde{p}^l(g)) + H(\tilde{p}(l)) \\ &= \sum_l \tilde{p}(l)\left[\sum_{i=1}^{l} H(\tilde{p}^l_i) - I(\tilde{p}^l)\right] + H(\tilde{p}(l)) \\ &= E_l\left\{\sum_{i=1}^{l} H(\tilde{p}^l_i)\right\} - E_l\left\{I(\tilde{p}^l)\right\} + H(\tilde{p}(l)). \end{aligned} \tag{13}$$

Combining Eqs. (12) and (13) we find

Lemma 1. *In the indirect encoding case, for any $p \in \Lambda$, any bijective encoding $\phi : G \to P$, and any factorized distribution $\tilde{q} \in \tilde{Q}$, we have*

$$D(p:q) = E_l\left\{I(\tilde{p}^l)\right\} + E_l\left\{D(\tilde{p}^l_{(1)} : \tilde{q}^l)\right\} + D(\tilde{p}(l):\tilde{q}(l)) \tag{14}$$

$$= E_l\left\{\sum_{i=1}^{l} H(\tilde{p}^l_i)\right\} + E_l\{D(\tilde{p}^l_{(1)} : \tilde{q}^l)\} + D(\tilde{p}(l):\tilde{q}(l)) + H(\tilde{p}(l)) - H(p). \tag{15}$$

The second RHS term in Eq. (15) is a comparison of only the marginals of $\tilde{p}^l$ and $\tilde{q}^l$, the third term is a comparison of the length distributions, the fourth term is the entropy of the genotype length, which depends on ϕ, and the last term depends only on p, not on ϕ or $\tilde{q}$. The first term in Eq. (15) is of particular interest here. The following bounds show how it relates to the description length. Note that $\log|\mathcal{A}|$ is the maximal entropy of a marginal

$$E_l\left\{\sum_{i=1}^{l} H(\tilde{p}^l_i)\right\} \leqslant \log|\mathcal{A}|\; E_l\left\{\sum_{i=1}^{l} 1\right\} = \log|\mathcal{A}|\; E_l\{l\} = L_p \log|\mathcal{A}|, \tag{16}$$

where we introduce $L_p = E_l\{l\}$ as the expected description length of samples of p in the encoding ϕ. On the other hand, from (13), we get

$$E_l\left\{\sum_{i=1}^{l} H(\tilde{p}_i^l)\right\} \geqslant H(p) - H(\tilde{p}(l)). \tag{17}$$

Note that for an optimally compact coding $L_p \log|\mathcal{A}| = H(p) - H(\tilde{p}(l))$,[6] and both bounds are tight.

We collect these findings in

Lemma 2. *In the indirect encoding case, the expected description length L_p of samples from p (parents) gives an upper bound on $D(p:q)$,*

$$\begin{aligned} 0 &\leqslant D(p:q) - E_l\{D(\tilde{p}_{(1)}^l : \tilde{q}^l)\} - D(\tilde{p}(l) : \tilde{q}(l)) \\ &\leqslant L_p \log|\mathcal{A}| - H(p) + H(\tilde{p}(l)). \end{aligned} \tag{18}$$

For an optimally compact encoding ϕ, these bounds are tight, i.e., we have

$$D(p:q) = E_l\{D(\tilde{p}_{(1)}^l : \tilde{q}^l)\} + D(\tilde{p}(l) : \tilde{q}(l)), \tag{19}$$

and $D(p:q)$ can easily be minimized by adapting the genotype marginals $\tilde{q}_i^l$ to estimate $\tilde{p}_i^l$ and setting the length distribution $\tilde{q}(l)$ equal to $\tilde{p}(l)$.

Let us briefly summarize and discuss these results by emphasizing certain aspects:

1. *Compression is doing "more than we need"*: Reconsider the exact expressions (14) and (15) of Lemma 1. We related the term $E_l\{\sum_i H(\tilde{p}_i^l)\}$ to the description L_p via the bound (16) and showed that this bound is exact for an optimal compression. One should note though that compression is not the only way to minimize the term $E_l\{\sum_i H(\tilde{p}_i^l)\}$; it can equally be minimized by reducing the marginal entropies $H(\tilde{p}_i^l)$, which means not to exploit the expressional power of the alphabet. This can better be understood going back to expression (14) involving the mutual information: Ultimately, what matters is to reduce $E_l\{I(\tilde{p}^l)\}$, i.e. finding a factorial code, which can also be done perfectly with very low marginal entropies, not exploiting the alphabet. By relating it to the description length we showed that a compression is reducing $E_l\{I(\tilde{p}^l)\}$ while *additionally* trying to exploit the alphabet optimally. Thus, compression is doing "more than we need" from the strict point of view of minimizing $D(p:q)$. Clearly, this also means that an optimally compact coding is not the only solution to minimize $D(p:q)$ via indirect induction—but it is one.

2. *If there are no further constraints on $\tilde{q}$* (e.g., no entropy bound) then $\tilde{q}^l$ and $\tilde{q}(l)$ can always be set equal to $\tilde{p}_{(1)}^l$ and $\tilde{p}(l)$, thus perfectly minimizing $E_l\{D(\tilde{p}_{(1)}^l : \tilde{q}^l)\} + D(\tilde{p}(l) : \tilde{q}(l))$. In this case, we have

$$D(p:q) = E_l\{I(\tilde{p}^l)\} = E_l\left\{\sum_{i=1}^{l} H(\tilde{p}_i^l)\right\} + H(\tilde{p}(l)) - H(p), \tag{20}$$

and the problem reduces to finding an encoding that extinguishes the mutual information $E_l\{I(\tilde{p}^l)\}$ or, as discussed, an optimal compression.

3. *The two step procedure*: If $\tilde{Q}$ is additionally constrained by a bound on the entropy, minimizing $D(p:q)$ remains a coupled problem (Eq. (14)) of reducing $E_l\{I(\tilde{p}^l)\}$ by a proper choice of ϕ and reducing $E_l\{D(\tilde{p}_{(1)}^l : \tilde{q}^l)\} + D(\tilde{p}(l) : \tilde{q}(l))$ by a proper choice of $\tilde{q}$, which though depends on ϕ. We discussed this coupled problem already in the fixed length case. The two step procedure of first finding a compact coding ϕ of p and then adapting the marginals of $\tilde{q}$ to those of $p \circ \phi$ is an approximate method for this minimization. In the worst case one may miss a reduction of $D(p:q)$

[6] Here is a slight difference to the usually considered case of channel capacity: In our case, the length of a genome itself can carry information (even for $\mathcal{A} = \{0\}$) of the amount $H(\tilde{p}(l))$ such that the symbols only need to encode $H(p) - H(\tilde{p}(l))$ entropy. In the channel capacity case, where a continuous stream of symbols is transmitted, the tight bound is $L_p \log|\mathcal{A}| = H(p)$, as for the Shannon–Fano code.

by an amount $\leqslant L_p\,\hbar$, when $\hbar$ is the lower bound on the entropy in each marginal $\tilde{q}_i^l$. Note that a compact coding minimizes this worst case error.

5. Compression EAs

5.1. General approach

A general approach to design a new EA, exploiting the results on compact representations, is to combine any compression technique with any standard EA. Such a *Compression EA* reads

(1) Initialize a finite population $q = \{x_1, \ldots, x_\lambda\} \subset P$.
(2) Evaluate q and select a subpopulation $p = \{x_1, \ldots, x_\mu\} \subset q$.
(3) Find a compression ϕ that (approximately) minimizes $L_p = \frac{1}{\mu}\sum_{i=1}^{\mu} \text{length}(\phi^{-1}(x_i))$.
(4) On the compressed representation, apply standard heuristic operators (e.g., mixing or EDA-operators) to generate λ offspring from μ parents.
(5) Map the λ offspring from G back to P using ϕ, yielding the new search samples q.
(6) Repeat from step 2.

The operators in step 4 may be any standard operators used in EAs. They have to be memory-less though, since the encoding will change in each iteration step and thus integrating knowledge from previous time steps may become futile.

One should emphasize that the crucial ingredient in a Compression EDA is the choice of the compression technique in step (3). This choice decisively determines the limitedness of the algorithm, i.e., kinds of problems it is able to solve. This is in strong analogy to the choice of the class of probabilistic models used in a traditional EDA. For instance, a simple model class like factorized distributions (PBIL) cannot express dependencies between solution variables and thus fails on problems with complex linkage, whereas more general classes of probabilistic models (e.g., Bayesian Networks, BOA) have a much wider range of problems they can be applied on. The same is true for different choices of compression techniques, although future research needs to clarify more specifically which kinds of compression techniques can express which kinds of dependency structures in problems.

In the remainder of this section, we present two Compression EAs. Both of these algorithms will use the same grammar-based compression technique, which clearly has limits w.r.t. the dependency structures it can capture. Thus, the presented algorithms should be understood as examples of Compression EAs. More specifically, the compression technique is limited in that it only detects dependencies between contiguous symbols and contiguous substrings of a string. This hierarchical notion of contiguity has its origin in the grammar-based compression and allows the induced model to capture dependencies that go beyond pairwise dependencies on only primitive symbols (i.e., beyond a simple Markovian model). Still, the technique is not versatile to capture arbitrary dependency structures (see also [27]).

The first of the presented Compression EAs is an EDA that is a direct implementation of the theoretical framework introduced in the previous sections and is strongly related to a variable length version of PBIL [2] on a compact representation. The second uses ordinary GA operators on the compact code. Before describing the grammar-based compression in Section 5.3 we address some heuristics to cope with finite sampling effect. We will test both algorithms on the Hierarchical XOR problem which aims to demonstrate their scalability up to binary strings of length 16 384.

5.2. Coping with stochasticity

When following the above sketched algorithm to design a Compression EDA, there are two specific points at which finite sampling and other stochastic effects lead to problems: (1) The optimal compression of a finite population $p = \{x_1, \ldots, x_\mu\}$ is typically "degenerate", by which we mean that it assigns a single symbol in $\mathcal{A} = \{1, \ldots, \mu\}$ to each sample. (2) Since the compression and thereby the model building is stochastic (steps 2 and 4 in Table 3) some generations might produce rather inferior offspring; a heuristic solution is that the statistics for model building "average" over a number of generations. We address these problems as follows.

5.2.1. Stochastic compression levels to avoid degenerate compression

Consider a finite number of samples $p = \{x_1, \ldots, x_\mu\}$. A mapping ϕ that maps these samples on an alphabet $\mathcal{A} = \{1, \ldots, \mu\}$ in the manner $x_i \mapsto i$ clearly is an optimal compression—with $L_p = 1$. We call this a *degenerate*

Table 2

Algorithm 1 (L-System Compression EDA (LC-EDA)).
parameters: sample, pool, and selection sizes λ [100], κ [1000], μ [100], entropy rate α [1]

(1) Initialize a population $q = \{x_1, \ldots, x_\lambda\} \subset P$ with random length 2 bit strings, and a *pool* $K = \emptyset \subset P$.
(2) Evaluate the samples in q.
(3) $K \leftarrow K \cup q$ (eliminate multiples), if $|K| > \kappa$ delete the oldest samples from K.
(4) Select a subset $p \subset K$ of the μ best samples in K.
(5) Calculate a compression ϕ for p (see table 3) yielding $\tilde{p} = p \circ \phi$.
(6) Calculate the length distribution $\tilde{p}(l)$, max length l^*, and marginals $\tilde{p}_i^l$ for $\tilde{p}$.
(7) Set $\tilde{q}(l) = (1 - \frac{\alpha}{l^*})\, \tilde{p}(l) + \frac{\alpha}{l^*}\, \mathcal{U}(\{1, \ldots, l^*+1\})$ and $\tilde{q}_i^l = (1 - \frac{\alpha}{l^*})\, \tilde{p}_i^l + \frac{\alpha}{l^*}\, \mathcal{U}(\mathcal{A})$, where $\mathcal{U}(\cdot)$ is the uniform distribution over a set.
(8) Take λ samples $\{g_1, \ldots, g_\lambda\}$ from the distribution $\tilde{q}$.
(9) Decode the samples via $x_i = \phi(g_i)$, giving a new set of search samples $q \leftarrow \{x_1, \ldots, x_\lambda\}$.
(10) Repeat from step 2.

compression. On the compressed representation the mutual information obviously vanishes. However, this representation is inapt as a basis for search. If the samples are disjoint, it is impossible to design a search distribution with $\tilde{q}(l) = \tilde{p}(l)$ that has more entropy than $\tilde{p}$ (which is $\log \mu$). The degenerate compression violates possible constraints on the lower bound on entropy of $\tilde{q}$.

A heuristic to solve this problem is to choose the level of compression stochastically. Since the L-System compression is an iterative procedure, this can be realized by escaping the compression loop at each iteration with a fixed probability β, see step 2 in Table 3.

5.2.2. Averaging statistics to build probabilistic models

To build probabilistic models reliably in a finite sampling algorithm one needs sufficient statistics. There are two simple alternatives to collect them. First, by choosing the sample size λ large enough so that the parent population in a single generation is large enough to build a model. Second, by averaging statistics in some way over a number of generations. For standard EDAs both approaches are in result approximately equivalent [10] and one should tend to favour the simpler first approach.

However, since our compression technique is in itself stochastic (steps 2 and 4 in Table 3) we prefer the second approach for the following reason. The offspring generated in a single generation can, due to the stochasticity of the model building, be inferior to previous generations and thus an unfortunate basis for model building, even when the sample size is chosen large enough. Averaging over generations helps to overcome this problem.

Standard averaging approaches in EDAs involve learning rates. For instance, in PBIL [2], when at generation t a new population of selected samples $p^{(t)} = \{x_1, \ldots, x_\mu\}$ is available and its marginals are $p_1^{(t)}, \ldots, p_l^{(t)}$, then the search distribution parameters (its marginals) are updated by $q_i^{(t)} = (1-\alpha)\, q_i^{(t-1)} + \alpha\, p_i^{(t)}$. Similarly, the pair-statistics A_{ij} in the COMIT algorithm [3], which are used to build the dependency tree model, are equally averaged by $A_{ij}^{(t)} = (1-\alpha)\, A_{ij}^{(t-1)} + \alpha\, B_{ij}^{(t)}$, where $B_{ij}^{(t)}$ is the pair-statistics of the current selected samples $p^{(t)}$.

We implement a similar heuristic in our Compression EDA—but a simple "decay" averaging of the distribution parameters is prohibitive because the representation changes in each generation and operators have to be memory-less. A simple solution is to store a larger pool $K = \{x_1, \ldots, x_\kappa\}$ of the last κ evaluated samples and use this as a basis to build the probabilistic model. The computational costs in space and time increase only by a constant factor. This is implemented in steps 3 and 4 in Table 2, providing a larger selected population p as a basis to build the probabilistic model.

The algorithm given in Table 2 describes the full L-System Compression EDA in detail.

5.3. Grammar-based compression

A simple technique of compression is to recursively analyze the parents for frequent pairs of neighboring symbols and replace such pairs by new symbols. We take this approach to compute a L-System compression following [17]

Table 3

Algorithm 2 (**L-System Compression**).
input: a set $p = \{x_1, \ldots, x_\lambda\}$ of integer sequences
output: the L-System Π (i.e., ϕ) and the compressed set p
parameters: the stochastic compression level parameter β [.1]

(1) Initialize the L-System $\Pi = \langle\rangle$ and the *new-symbol* $c = 1 + \max\{\text{integers in p}\}$.
(2) With a probability β, exit the algorithm.
(3) Calculate the frequency of every symbol pair that occurs in p.
(4) For every pair $r_1 r_2$ of symbols that occurs more often than once, create a new production $c \to r_1 r_2$, append it at the *beginning* of Π, and increment the *new*-symbol $c \leftarrow c + 1$. This is to be done in order, starting with the pair of highest frequency, and random order between pairs of equal frequency. If there are no such pairs, exit the algorithm.
(5) Recode the population $p \leftarrow \{\phi^{-1}(x_1), \ldots, \phi^{-1}(x_\mu)\}$ (effectively, only the new productions will result in replacements).
(6) Repeat from step 2.

Table 4
An L-System found to compress a population of identical solutions to the 1024-bit HXOR

C:10	K:0110 1001 1001 0110
D:110	L:0110 1001 0110
E:0110	M:0110 1001 0110 1001 1001 0110
F:1 0110	N:1001 0110 0110 1001 1001 0110
G:01 0110	O:1001 0110 0110 1001 1001 0110 0110 1001 0110 1001 1001 0110
H:01 1001 0110	P:1001 0110 0110 1001 0110 1001 1001 0110
I:1001 1001 0110	*etc...*
J:1001 0110	

The full L-System is composed of 27 productions and reads ⟨C:10, D:1C, E:0D, F:CD, G:0F, H:EG, I:CH, J:CG, K:EI, L:EJ, M:LI, N:JK, O:NM, P:JM, Q:KO, R:PO, S:KR, T:QS, U:QP, V:UT, W:SV, X:US, Y:PT, Z:PW, a:XY, b:VZ, c:ba⟩. In the table, we expanded the RHS of these productions; for brevity only the first 15 productions are displayed. The single symbol c expands to the length 1024 solution.

(see also [15] for a brief review of grammar-based compression). An L-System is a sequence $\Pi = \langle \pi_1, \ldots, \pi_k \rangle$ of k productions π_i. Given the alphabet $\mathcal{A}$, each production $\pi = (l \to r_1 \ldots r_m)$ consists of a LHS symbol $l \in \mathcal{A}$ and a sequence $r_1 \ldots r_m$ of RHS symbols. The L-System Π defines a mapping ϕ from one sequence x to another by applying all productions π_i, in the given order, on x. Applying a production $l \to r_1 \ldots r_m$ on x means to replace every l that occurs in x by the sequence $r_1 \ldots r_m$. The mapping ϕ can be inverted by applying all productions, in reverse order, inversely on a sequence. Inverse application of a production $l \to r_1 \ldots r_m$ on x means to replace every subsequence $r_1 \ldots r_m$ that occurs in x by l.

Let $\mathcal{A}$ be the non-negative integer numbers, $\mathcal{A} = \mathbb{N}_0$. Starting with a population $p = \{x_1, \ldots, x_\lambda\}$ of integer sequences, there is a straight-forward way to construct an L-System that compresses the population by recursively extracting and encapsulating pairs of symbols that occur frequently in the population. Table 3 describes this scheme; the computational complexity of constructing such an L-System is only linear in the string length when using certain efficient data structures [17].

To give an impression on the codings developed by the L-System Compression scheme, Table 4 displays an L-System found to compress a population of identical solutions to the 1024-bit HXOR problem (see Section 5.4). In this case, we did not exit the compression loop at some stochastic level but calculated the full, degenerate compression such that eventually every sample is represented by a single symbol c. We find that some productions represent the typical modules of HXOR solutions (like E:0110 or J:1001 0110) while others represent parts of these modules. Recall that the order in which productions are added to the L-System is stochastic (cf. step 4 in Table 3). Consequently, the coding is not strictly hierarchical as a human might have designed it (or DEVREP, see Section 5.5) and the L-System comprises more pair-productions than minimally necessary to encode the sequence of length 1024 (the minimum would be 19 productions).

5.4. The Hierarchical XOR problem

The fitness function we consider is the Hierarchical XOR (HXOR) function [29,8]. For a string $x \in \{0,1\}^n$ we first define a boolean function $h(x) \in \{0,1\}$, determining whether x is "valid" or not: Let

$$n = \text{length}(x), \quad l = \lfloor \log_2(n-1) \rfloor \in \mathbb{N}_0, \quad L = x_{1:2^l}, \quad R = x_{2^l+1:n}. \tag{21}$$

Note that l is an integer cutting the string into a left part L and a right part R at position 2^l. We define

$$h(x) = \begin{cases} 1 & \text{if } n = 1, \\ 1 & \text{if } n = 2^l \ \wedge \ h(L) = 1 \ \wedge \ h(R) = 1 \ \wedge \ L = \overline{R}, \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

Here $\overline{R}$ is the bit-wise negation of the right part. The last condition $L = \overline{R}$ means that the bit-wise XOR between left and right part must be true for each bit. For instance, the strings for which $h(x) = 1$ are, up to length 16, $\langle 0 \rangle$, $\langle 01 \rangle$, $\langle 0110 \rangle$, $\langle 0110\,1001 \rangle$, $\langle 0110\,1001\,1001\,0110 \rangle$, and their bit-wise negations. Based on h, we define a fitness function $H(x) \in \mathbb{N}$, for $n \geqslant 2$,

$$H(x) = H(L) + H(R) + \begin{cases} n & \text{if } h(x) = 1, \\ 0 & \text{else} \end{cases} \tag{23}$$

and $H(x) = 1$ if $n = 1$. To normalize and put a limit on the string length, we define the lth HXOR function $H_l(x) \in [0,1]$: If x is longer than 2^l, let $x' = x_{1:2^l}$ and otherwise $x' = x$. Then,

$$H_l(x) = \frac{1}{2^l(l+1)} H(s'). \tag{24}$$

The normalization is given by the highest possible value $2^l(1+l)$ of $H(x)$ for a length 2^l string. There exist two global optima of H_l, namely the two "valid" strings of length l for which $h(x) = 1$.

5.5. Results of the L-System Compression EDA

We tested the L-System Compression EDA by running 20 trials on each of the HXOR problems of lengths $\{2^1, 2^2, \ldots, 2^{14} = 16\,384\}$. For all problem sizes we chose the same parameter settings of algorithm 1 and 2 as given in the brackets in Tables 2 and 3. Clearly, the optimal choice of parameters depends on the problem size; we chose the same parameter settings for all problem sizes only for simplicity. As a cost measure we investigated the number of string evaluations and the number of bit evaluations (assuming the cost of each string evaluation were equal to its length) needed to find a solution, the latter to allow comparison with [8].

All runs consistently found a solution. Fig. 1 displays the average costs needed together with the standard deviations. Generally, the performance of the algorithm—in terms of the generations needed—is of the same order as the DEVREP algorithm presented by [8], which is the only previous algorithm we are aware of capable of solving large HXOR problems. (de Jong [8] reported only one result for the 64- and 1024-bit HXOR problem, where about 2.3×10^7 bit evaluations were needed in the single 1024-bit run presented.) The DEVREP algorithm is tailored to hierarchical problems, where different hierarchy levels are explicitly distinguished and mutational variations allowed only within a specific hierarchy level. It should be clear that plain variable length GAs fail on (reasonable length) HXOR problem because of the hierarchy of local minima when search is performed on a direct representation (see [8] for experiments).

Fig. 1 also displays polynomials that are fitted to the performance of the LC-EDA in figures based on the means and variations of the performance considering only problems of size 64 or larger. These suggest that the number of evaluations scales with $l^{0.534 \pm 0.010}$ when l is the size of the problem (in bits) while the number of bit evaluations scales with $l^{1.564 \pm 0.012}$.

5.6. A L-System Compression GA and results

GAs are heuristic search schemes, and as such they can be understood by investigating what kind of search distribution they induce depending on previously evaluated samples (the parent population). A traditional GA has

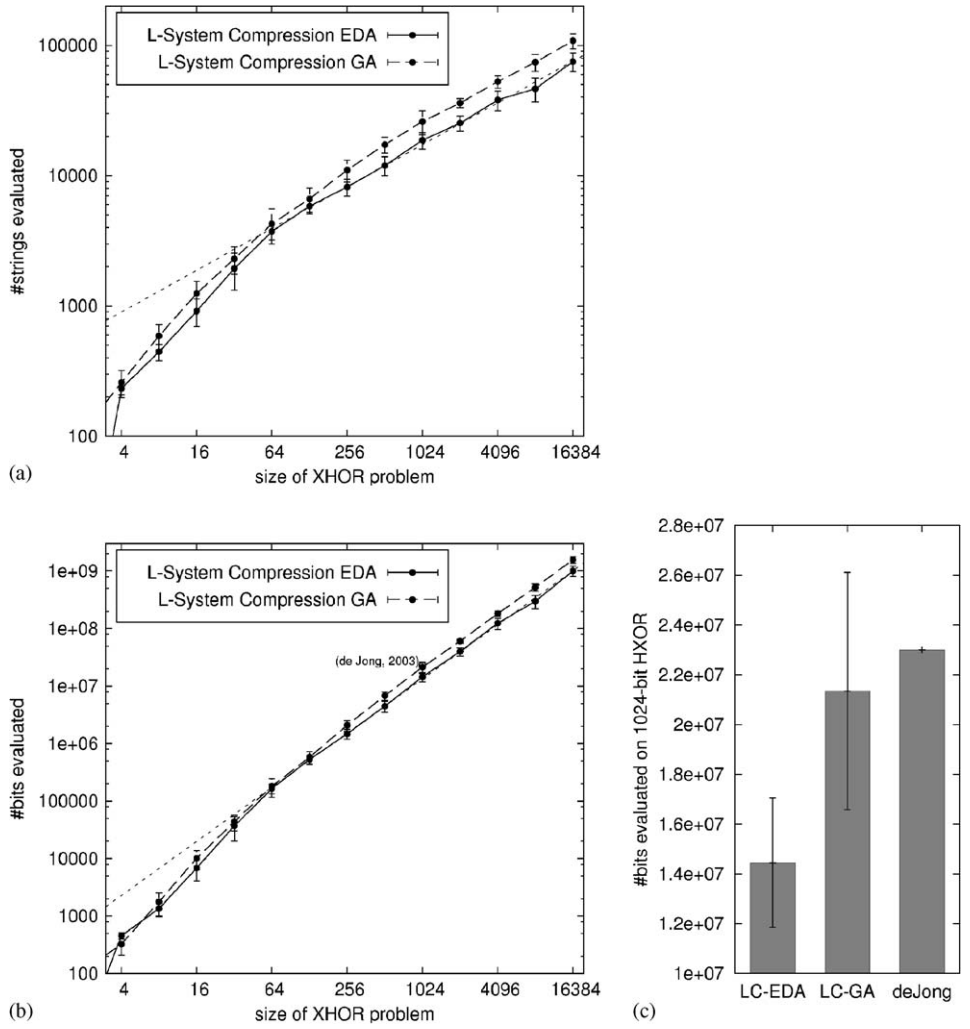

Fig. 1. The number of (a) string evaluations and (b) bit evaluations needed to find an optimum for HXOR problems of lengths $\{2^1, 2^2, \ldots, 2^{14}\}$. For each problem size we collected 20 independent runs and display here the average and standard deviation. The dotted line are polynomials: (a) $y \propto x^{0.534\pm0.010}$ and (b) $y \propto x^{1.564\pm0.012}$ fitted to the performance of the LC-EDA for problem sizes $\geqslant 64$. (c) more clearly than the log-plots displays the differences in performance for 1024-bit HXOR problem.

independent symbol alterations as a mutation operator, which induces a factorized variability distribution σ for each individual. Considering the whole parent population, a mutation only GA has a mixture $q = \frac{1}{\mu}\sum_{i=1}^{\mu}\sigma_i$ of factorized distributions as search distribution. Crossover is an operator that makes this search distribution "more factorized". Actually, one can understand crossover as a move in distribution space from the parent population to a more factorized version of the parent population without changing the gene marginals. One can show that both, mutation and crossover, can only reduce the total mutual information that was present in the parent population [26]. In any case, although GAs do not induce a perfectly factorized search distribution, its inherent search mechanisms are of a factorial nature. This is also the reason why they have been modeled as factorial search, e.g., by PBIL or gene pool GAs.

We also implemented a L-System Compression GA, which uses a standard crossover and mutation operator to generate offspring on the compact representation in place of building a probabilistic model. The exact algorithm is given in Table 5, the results also in Fig. 1. The mutation operator applies ξ mutations on a sample, where ξ is drawn from a Poisson distribution with mean 1. Possible mutations are (1) replacing, (2) inserting, or (3) deleting a random symbol

Table 5

Algorithm 3 (**L-System Compression GA** (**LC-GA**)).
parameters: sample and selection sizes λ [100], μ [30]

(1) Initialize a population $q = \{x_1, \ldots, x_\lambda\} \subset P$ with random length 2 bit strings.
(2) Evaluate the samples in q.
(3) Select the subset $p \subset q$ of the μ best samples in q.
(4) Calculate a compression ϕ for p (see table 3) yielding $\tilde{p} = p \circ \phi$.
(5) Apply crossover and mutations (as detailed in the text) on $\tilde{p}$ to generate λ new offspring $\tilde{q}$.
(6) Decode the offspring $\tilde{q}$ to q.
(7) Add an elitist (the best of p) to q.
(8) Repeat from step 2.

from/in the sample; each of these applies with equal probability. The crossover operator splits the parent population randomly into pairs and for each pair applies uniform crossover to generate two offspring. If one parent is longer than the other, one offspring inherits this extra sequence, the other doesnot.

6. Discussion

Beyond the introduction of Compression EAs, the main goal of this paper was to provide a theoretical grounding for the use of compact representations for stochastic search. The main results, subsumed in Lemmas 1 and 2, establish the relationship between the Kullback–Leibler divergence (KLD) and the description length of a variable-length encoding. Namely, minimizing the KLD between a selected distribution p and a search distribution q can be achieved by finding a representation that minimizes the description length of p-samples. Then a simple factorized search distribution $\tilde{q}$ on the compact representation allows us to model $\tilde{p}$.

This result is one piece of a boarder perspective on the role of factorial representations for search. Generally, finding a factorial representation for data to a large extent means to understand structural characteristics of the data. Independent Component Analysis [14] and other latent variable models that try to construct factorial representations demonstrate the depth of this concept in the general Machine Learning context. In [27], various possibilities to construct factorial representations (in a concrete variable length string context) are discussed. The representations proposed there typically achieve independence between variables by *expanding* the representation. Finding factorial representations by compression seems a comparably unique approach.

However, the choice of a specific compression technique is crucial for a compression-based search algorithm and decides on whether structural patterns in the data can be detected and exploited for further search. In fact, the problem of compression is strongly related to that of probabilistic modeling; any compression technique first needs to analyze dependencies in the data before it can introduce new symbols for dependent features. Hence, it is a matter of the choice of compression technique whether a factorial representation can be found for a specific type of problem and the choice of compression technique is analogous to the choice of class of probabilistic models in traditional EDAs. Standard string compression algorithms, such as Lempel–Ziv compression as well as the L-System compression we investigated here, are limited in that they basically search for contiguous patterns in the original or partly compressed string. This was successful for the HXOR problem since the dependencies can (on various hierarchy levels) be described in terms of contiguous patterns. For many other hard optimization problems (e.g., MAXSAT) the dependencies will typically be between arbitrary variables and hardly detectable for standard string compression techniques. In its generality, we cannot answer the question of what kind of compression techniques are suited for a specific class of problems. More general analysis of dependencies between variables (as sketched in [27]) might lead a way to more general compression approaches for search.

The results presented here also have implications for the understanding of natural evolution. In Section 5.6, we briefly explained why standard crossover and mutation operators in GAs are comparable to a factorized search distribution. At least approximately, also in nature variability is generated by independent mutational incidents. So, assuming that nature must rely on roughly factorial search distributions, evolution would certainly profit from finding a factorial representation of solutions (organisms). This discussion might seem rather artificial—how could evolution construct

itself factorial representations of the distribution of previously successful solutions? However, accounting for the possibility of neutral evolution [13], this becomes very plausible.

Neutrality means that the genotype–phenotype mapping $\phi : G \to P$ is non-injective (i.e., many-to-one). Many different genetic representations $g \in [x]_\phi$ exist to encode the same phenotype x and neutral mutations can transform one representation g_1 to another representation g_2 of the same phenotype. It was shown that selection also induces an implicit selectional pressure on the choice of representation of a phenotype, which leads to a directed evolution also within the sets $[x]_\phi$ (neutral evolution or drift). This selectional pressure turns out to be proportional to the (negative) KLD (which is related to the effective fitness [18,23]; see also the discussions in [25] and [6]).

So, the central term that was discussed throughout this paper also determines the evolution of genetic representations in the case of neutrality. When here we proved that minimization of the KLD is related to finding a compact representation, this implies that the implicit selection pressure tends to select compact representations within a neutral set. In fact, in [24] genotypes were considered to be L-Systems and neutral mutations could transform one L-System into an equivalent one (which expresses to the same phenotype). The underlying selectional pressure on the genetic representation indeed lead to compact L-Systems to describe phenotypes. There, it was argued that the origin of compactness is the advantage in mutational robustness when every genetic symbol underlies a constant mutation rate. The results derived here additionally imply that compact representations are *structurally* more favorable—the phenotypic variability they induce tends to minimize the KLD and follow the Estimation-of-Distribution search heuristic.

Acknowledgments

I would like to thank Chris Williams for the inspiring discussions on compression and latent variable probabilistic models, and the German Research Foundation (DFG) for their funding of the Emmy Noether fellowship TO 409/1-1, allowing me to pursue this research.

References

[1] S. Amari, Information geometry on hierarchy of probability distributions, IEEE Trans. Inform. Theory 47 (2001) 1701–1711.
[2] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, Technical Report CMU-CS-94-163, Computer Science Department, Carnegie Mellon University, 1994.
[3] S. Baluja, S. Davies, Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space, in: Proc. 14th Int. Conf. on Machine Learning, ICML 1997, 1997, pp. 30–38.
[4] L. Barbulescu, J.-P. Watson, D. Whitley, Dynamic representations and escaping local optima: improving genetic algorithms and local search, in: 17th National Conf. on Artificial Intelligence (AAAI), 2000, pp. 879–884.
[5] A. Barron, J. Rissanen, B. Yu, The minimum description length principle in coding and modeling, IEEE Trans. Inform. Theory 44 (1998) 2743–2760.
[6] K. Burjorjee, J. Pollack, Theme preservation and the evolution of representation, in: Genetic and Evolutionary Computation Conference (GECCO 2005); Theory of Representations Workshop, 2005.
[7] J.S. de Bonet, C.L. Isbell Jr., P. Viola, MIMIC: finding optima by estimating probability densities, in: M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems, Vol. 9, The MIT Press, Cambridge, MA, 1997, p. 424.
[8] E.D. de Jong, Representation development from Pareto-Coevolution, in: Genetic and Evolutionary Computation Conference, GECCO 2003, 2003, pp. 265–276.
[9] G. Harik, Linkage learning via probabilistic modeling in the ECGA, Technical Report IlliGAL-97006, Illinois Genetic Algorithms Laborator, 1999.
[10] G. Harik, F. Lobo, D. Goldberg, The compact genetic algorithm, Technical Report IlliGAL-97006, Illinois Genetic Algorithms Laborator, 1997.
[11] R.B. Heckendorn, A.H. Wright, Efficient linkage discovery by limited probing, Evolutionary Comput. 12 (2004) 517–545.
[12] G.S. Hornby, J.B. Pollack, The advantages of generative grammatical encodings for physical design, in: Proc. 2001 Cong. on Evolutionary Computation, CEC 2001, IEEE Press, New York, 2001, pp. 600–607.
[13] M.A. Huynen, Exploring phenotype space through neutral evolution, J. Molecular Evol. 43 (1996) 165–169.
[14] A. Hyvärinen, E. Oja, Independent component analysis: algorithms and applications, Neural Networks 13 (2000) 411–430.
[15] E. Lehman, A. Shelat, Approximation algorithms for grammar-based compression, in: Proc. 13th Annu. ACM-SIAM Symposium on Discrete Algorithms, SODA 2002, ACM/SIAM, 2002, pp. 205–212.
[16] G.E. Liepins, M.D. Vose, Representation issues in genetic algorithms, J. Experimental Theoret. Artif. Intell. 2 (1990).
[17] C.G. Nevill-Manning, I.H. Witten, Identifying hierarchical structure in sequences: a linear-time algorithm, J. Artif. Intell. Res. 7 (1997) 67–82.
[18] P. Nordin, W. Banzhaf, Complexity compression and evolution, in: L. Eshelman (Ed.), Genetic Algorithms: Proc. Sixth Internat. Conf., ICGA 1995, Morgan Kaufmann, Pittsburgh, 1995, pp. 310–317.
[19] M. Pelikan, D.E. Goldberg, Hierarchical BOA solves Ising spin glasses and MAXSAT, in: Genetic and Evolutionary Computation Conf. (GECCO 2003), Springer, Berlin, 2003, pp. 1271–1282.

[20] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, Linkage problem, distribution estimation, and Bayesian networks, Evolution. Comput. 9 (2000) 311–340.
[21] M. Pelikan, D.E. Goldberg, F. Lobo, A survey of optimization by building and using probabilistic models, Technical Report IlliGAL-99018, Illinois Genetic Algorithms Laboratory, 1999.
[22] F. Rothlauf, D.E. Goldberg, Redundant representations in evolutionary computation, Evolution. Comput. 11 (2003) 381–415.
[23] C.R. Stephens, J.M. Vargas, Effective fitness as an alternative paradigm for evolutionary computation I: general formalism, Genetic Programm. Evolvable Mach. 1 (2000) 363–378.
[24] M. Toussaint, Demonstrating the evolution of complex genetic representations: an evolution of artificial plants, in: Genetic and Evolutionary Comput. Conf., GECCO 2003, 2003, pp. 86–97.
[25] M. Toussaint, On the evolution of phenotypic exploration distributions, in: C. Cotta, K. De Jong, R. Poli, J. Rowe (Eds.), Foundations of Genetic Algorithms 7 (FOGA VII), Morgan Kaufmann, 2003, pp. 169–182.
[26] M. Toussaint, The structure of evolutionary exploration: on crossover, buildings blocks, and Estimation-of-Distribution algorithms, in: Genetic and Evolutionary Comput. Conf., GECCO 2003, 2003, pp. 1444–1456.
[27] M. Toussaint, Factorial representations to generate arbitrary search distributions, in: Genetic and Evolutionary Computation Conference, GECCO 2005; Theory of Representations workshop, ACM Press, Washington, DC, USA, 2005, pp. 339–345.
[28] P.M.B. Vitányi, M. Li, Minimum description length induction, Bayesianism, and Kolmogorov complexity, IEEE Trans. Inform. Theory IT-46 (2000) 446–464.
[29] R.A. Watson, J.B. Pollack, Hierarchically consistent test problems for genetic algorithms: summary and additional results, in: Late breaking Papers at the Genetic and Evolutionary Comput. Conf., 1999, pp. 292–297.
[30] D. Whitley, S. Rana, R. Heckendorn, Representation issues in neighborhood search and evolutionary algorithms, in: Genetic Algorithms and Evolution Strategy in Engineering and Computer Science, Wiley, New York, 1997, pp. 39–58.
[31] A.H. Wright, R. Poli, C.R. Stephens, W.B. Langdon, S. Pulavarty, An estimation of distribution algorithm based on maximum entropy, in: Genetic and Evolutionary Computation Conference, GECCO 2004, Springer, Berlin, 2004, pp. 343–354.