Compact genetic codes as a search strategy of evolutionary processes

Marc Toussaint

Institute for Adaptive and Neural Computation, University of Edinburgh,
5 Forrest Hill, Edinburgh EH1 2QL, Scotland, UK
mtoussai@inf.ed.ac.uk

Abstract. The choice of genetic representation crucially determines the capability of evolutionary processes to find complex solutions in which many variables interact. The question is how good genetic 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 variability 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 variability can be characterized as minimizing the Kullback-Leibler divergence to a population of previously selected individuals (parents). Our core result is a relation between the Kullback-Leibler divergence and the description length of the encoding in the specific scenario, stating that compact codes provide a way to minimize this divergence. A proposed class of Compression Evolutionary Algorithms and preliminary experiments with an L-system compression scheme illustrate the approach. We also discuss the implications for the self-adaptive evolution of genetic representations on the basis of neutrality (σ-evolution) towards compact codes.

1 Introduction

The complexity of a problem largely depends on the interactions between variables of the solution. A stochastic search process like evolution 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. The ability of natural evolution to find highly complex structured organisms, in which many variables interact in determining the fitness, can only be understood when acknowledging that evolution did not pursue an exhaustive search but “learned” to shape the search distribution towards complex, highly structured organisms (see also [25]).

In evolutionary processes the search distribution is determined by the variational operators (mutation and recombination) and, in the case of indirect encodings, by the choice of genetic representation. The role of the genetic representation seems to become particularly important when large and regularly or hierarchically structured solutions need to be found. The eye-less gene [8] is an impressive example of a genetic representation specifically designed to induce modular structured variability in nature.

The general questions are how to find genetic representations that induce the desired dependencies on the phenotype 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 [12, 27, 4, 18, 11]. Theoretical approaches concerning the adaptation of the representation based on the specific current population include, for example, Estimation-Of-Distribution Algorithms [17], Walsh analysis [10], and Maximal Entropy principles [28].

Our approach is to consider a specific scenario that we term indirect Estimation-of-Distribution: We assume that the search distribution (mutational variability) on a variable length genotype space is decorrelated. A one-to-one encoding onto the phenotype space then needs to induce a proper structured phenotypic variability. 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 variability can be characterized as minimizing the Kullback-Leibler divergence to a population of previously selected individuals (parents). Our core result is a relation between the Kullback-Leibler divergence 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 genetic codes 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, 24]) to the specific domain of evolutionary search based on adaptive representations.

An intuitive way to grasp these results might be the following: Consider a set of good (selected) individuals in the phenotype space. In this parent population there will generally exist dependencies between phenotypic variables of the individuals, measurable as mutual information between them. This information on the dependencies—that stem from selection and have their origin in the structure of the problem—is what should be extracted and exploited for further search. Assume we can map these individuals 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 symbols that describe the individuals because otherwise the mapping would not be a minimum description length compression. Thus, a compression is one way (among others) to map on genotypic representations in which symbols are decoupled (i.e., to map on a factorial code). The 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. In other words, when there is noise (decorrelated mutational variability) on the genetic symbols, this noise should translate to phenotypic noise that obeys these dependencies.

The following two sections will introduce the theoretical framework, including the indirect induction of a search distribution and the principle of Estimation-of-Distribution Algorithms. Section 4 derives the main results on the relation between compact codes and Estimation-of-Distribution. Section 5 aims to illustrate the approach by proposing a class of Compression Evolutionary Algorithms and presenting preliminary experiments with a Compression GA based on a simple L-system compression scheme (similar to the ideas of Nevill-Manning and Witten [13]). This leads to the discussion, summarizing the main results, but also considering practical issues on accumulative versus each-time-step compression, and implications for understanding the fully self-adaptive evolution of genetic representations on the basis of neutrality (σ-evolution) towards compact codes.

2 Indirect induction of search distributions

Let P be the search space. A heuristic search scheme is a process in the space Λ^P of distributions over the search space in which a search distribution $q \in \Lambda^P$ 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. This 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.

In this paper we are interested in indirect codings of search points. In the heuristic search framework, this means that a distribution $\tilde{q} \in \Lambda^G$ over another search 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*

$\phi:\ G \to P$,

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

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.

Two additional constraints define the "indirect encoding case" considered in this paper: First, G is the space of variable length strings over some finite alphabet $\mathcal{A}$,

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

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

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

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 $\tilde q_l^i$ over the ith symbol ($i \le l$) and the mutual information $I(\tilde q_l) = \sum_i H(\tilde q_l^i) - H(\tilde q_l)$ can then be defined as usual.[1]

As the second constraint we limit the space of possible search distributions in a certain way: 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. And we impose that the length-conditioned distributions $\tilde q_l$ on the genotype space have to factorize. We denote the set of feasible genotype distributions by

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

Putting both together, we have the space $Q \subseteq \Lambda^P$ of feasible search distribution over P as

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

In summary, indirect induction of the search distribution means that, in order to design a search distribution $q \in \Lambda^P$ we have to pick a coding $\phi \in \Phi$ and a decorrelated distribution $\tilde q \in \tilde Q$ on the genotype space. In other terms, every feasible search distributions q correspond to a pair $(\phi, \tilde q)$.

[1] For a distribution q over some product space $\mathcal{A} \times \cdots \mathcal{A}$, 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).

3 Estimation-Of-Distribution

What is a reasonable heuristic to design a search distribution given the results of evaluation of previous sample points? We will follow here the idea of Estimation-Of-Distribution Algorithms [17] which can be described as follows.

We assume that the outcome of evaluation is given as a distribution p over P, typically the parent population.[2] Given that the space of feasible search distributions is limited to Q, a simple heuristic to chose the new search distribution is to pick the one that is most similar to p. Similarity can be measured by the Kullback-Leibler divergence (KLD) $D\big(p:q\big) = \mathrm{E}_p\left\{\log\frac{p(x)}{q(x)}\right\}$ between two distributions, which also captures the structural similarity between two distributions in the sense of the similarity of the different order dependencies (see [1, 23], please note the relations between the KLD, log-likelihood, free energy, and mean energy[3]). Thus, the heuristic to design the new search distribution q' reads

$$q' = \operatorname*{argmin}_{q\in Q} D\big(p:q\big) \ . \tag{2}$$

We term this specific kind of an EDA *KL-search*. 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 (2) in every step. On the other hand, it determines the algorithm's capability to exploit the structure observable in p.

Some algorithms of the class of EDAs are exact instantiations of KL-search: MIMIC [6] 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 [16] 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 [3], Bayesian networks (BOA, [16]), or Bayesian networks with decision trees at each node (hBOA, [15]).

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

[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 "fitness distribution over offspring", which corresponds to the parent population.

[3] With the definitions of the entropy $H(p) = \mathrm{E}_p\{\log p(x)\}$ and the log-likelihood $\mathcal{L}(q) = \mathrm{E}_p\{\log q(x)\}$ we have $D\big(p:q\big) = -H(p) - \mathcal{L}(q)$. One could roughly say, "minimizing the KLD means maximizing the log-likelihood *and* the entropy". Further, when defining an energy functional $E(x) = -\log q(x)$ and considering its Boltzmann distribution $p(x) = \frac{1}{Z}\,\epsilon^{-E(x)}$, then the mean energy $E = \mathrm{E}_p\{E(x)\} = -\mathcal{L}(q)$ is the negative log-likelihood while the free energy $F = -\ln Z = E - H(p)$ is the KLD.

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.

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 indirect coding case, a choice of q means to pick a coding ϕ and a factorized distribution $\tilde q$ on G. Putting this together, KL-search amounts to a heuristic to pick a coding ϕ 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. Let us first 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 $I(\tilde q) = \sum_i H(\tilde q^i) - H(\tilde q) \overset{!}{=} 0$ constrains $\tilde q$ to vanishing dependencies (of arbitrary order) between genes.

Combining (2) with the definition (1) of Q for the case of indirect codings, we find

$$D\big(p : q\big) = \sum_x p(x) \ln \frac{p(x)}{\sum_{g' \in [x]_\phi} \tilde q(g')} = \sum_g \tilde p(g) \ln \frac{\tilde p(g)}{\tilde q(g)} \ . \tag{3}$$

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 has to be bijective such that there exists exactly one $g \in [x]_\phi$. Next we use that $\tilde q$ has to factorize, $\tilde q(g) = \tilde q(g^1, g^2, ..., g^l) = \tilde q^1(g^1)\ \tilde q^2(g^2) \cdots \tilde q^l(g^l)$,

$$\begin{aligned} D\big(p : q\big) &= \sum_g \tilde p(g) \ln \frac{\tilde p(g)}{\tilde p^1(g^1) \cdots \tilde p^l(g^l)} + \sum_g \tilde p(g) \ln \frac{\tilde p^1(g^1) \cdots \tilde p^l(g^l)}{\tilde q^1(g^1) \cdots \tilde q^l(g^l)} \\ &= I(\tilde p) + D\big(\tilde p^{(1)} : \tilde q\big) \ . \end{aligned} \tag{4}$$

Here we defined $\tilde p^{(1)}(g) = \tilde p^1(g^1)\ \tilde p^2(g^2) \cdots \tilde p^l(g^l)$ as the "factorized reduction" of $\tilde p$ (i.e, the product of its marginals, see also [1, 23] on how to generally define the kth order reduction $\tilde p^{(k)}$ of $\tilde p$ containing all and only the dependencies of order $\le k$ within $\tilde p$).

This result states that, in order to follow the KL-search scheme in the indirect coding case, one should find a coding ϕ and a search distribution $\tilde q$ such that $I(\tilde p) + D\big(\tilde p^{(1)} : \tilde q\big)$ 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. In this case, no matter which ϕ is chosen, one can always minimize $D\big(\tilde p^{(1)} : \tilde q\big)$ down to zero by picking $\tilde q = \tilde p^{(1)}$. Thus, since $I(\tilde p)$ is independent of $\tilde q$, the minimization (2) can be realized by first optimizing ϕ and then picking $\tilde q$:

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

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. (Recall that for $\tilde q = \tilde p^{(1)}$, and since ϕ is bijective, $H(p) = H(\tilde p) = H(\tilde p^{(1)}) - I(P_G) = H(\tilde q) - I(\tilde p) = H(q) - I(\tilde p)$. Therefore, the entropy of search $H(q)$ would only be greater than $H(p)$ by the amount of $I(\tilde p)$, which is minimized.) Hence, in the second case, when $\tilde Q$ is only a subset of all factorized distributions, $D\big(\tilde p^{(1)} : \tilde q\big)$ can generally not be minimized to zero and the minimization of (2) 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 $D\big(\tilde p^{(1)} : \tilde q\big)$ w.r.t. $\tilde q$. The exact minimization of (2) 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 (2), and let $(\phi', \tilde q')$ be the result of the two step procedure, i.e., ϕ' minimizes $I(\tilde p')$ and $\tilde q'$ minimizes $D\big(\tilde p'^{(1)} : \tilde q'\big)$ 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,

$$\begin{aligned} D\big(p : q'\big) - D\big(p : q^*\big) &= D\big(\tilde p'^{(1)} : \tilde q'\big) - D\big(\tilde p^{*(1)} : \tilde q^*\big) + I(\tilde p') - I(\tilde p^*) \\ &\le D\big(\tilde p'^{(1)} : \tilde q'\big) - D\big(\tilde p^{*(1)} : \tilde q^*\big) \ \le \ D\big(\tilde p'^{(1)} : \tilde q'\big) \\ &= \sum_{i=1}^{l} D\big(\tilde p'^{i} : \tilde q'^{i}\big) \\ &\le -l \, \log \tilde q'^{i}(a^i) \ = \ -l \, \log(1-\alpha) \ \le \ l \, \hbar \ . \end{aligned}$$

The first inequality stems from the fact that ϕ' minimizes $I(\tilde p')$ and thus $I(\tilde p') \le I(\tilde p^*)$. Since both, $\tilde p'^{(1)}$ and $\tilde q'$, are factorized distributions, their Kullback-Leibler divergence decomposes into a sum. For each marginal, when there is a lower bound $\hbar$ on the entropy of $\tilde q'^{i}$, the divergence $D\big(\tilde p'^{i} : \tilde q'^{i}\big)$ is particularly large when $\tilde p'^{i}$ has very low entropy. In the worst case, $\tilde p'^{i}$ has zero entropy, i.e., is nonzero only for a single symbol $a^i \in \mathcal{A}$. In that case $D\big(\tilde p'^{i} : \tilde q'^{i}\big) = -\log \tilde q'^{i}(a^i)$. In order to minimize $D\big(\tilde p'^{i} : \tilde q'^{i}\big)$, $\tilde q'^{i}$ is chosen to have the form of the typical symbol mutation distribution with mutation rate α, where $\tilde q'^{i}(a) = 1-\alpha$ for $a = a^i$ and $\tilde q'^{i}(a) = \frac{\alpha}{|\mathcal{A}|-1}$ for $a \neq a^i$. Then, $H(\tilde q^i) = -(1-\alpha)\log(1-\alpha) - \alpha \log \frac{\alpha}{|\mathcal{A}|-1}$. Given the lower bound $\hbar$ on the entropy, the minimal mutation rate α can be chosen to ensure $H(\tilde q^i) = \hbar$ and $D\big(\tilde p^i : \tilde q^i\big) = -\log(1-\alpha) \le \hbar$. Thus, in the worst case, the "error" made when using the two step procedure instead of the exact minimization of (2) is smaller than $l \, \hbar$.

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 $l = \text{length(g)}$.) Equation (3) now leads to

$$
\begin{aligned}
D\big(p :q\big) &= \sum_l \sum_g \tilde{p}_l(g)\tilde{p}(l) \ln \frac{\tilde{p}_l(g)\tilde{p}(l)}{\tilde{q}_l(g)\tilde{q}(l)} \\
&= \sum_l \tilde{p}(l) \sum_g \tilde{p}_l(g) \ln \frac{\tilde{p}_l(g)}{\tilde{q}_l(g)} + \sum_l \tilde{p}(l) \ln \frac{\tilde{p}(l)}{\tilde{q}(l)} \\
&= \sum_l \tilde{p}(l) \sum_g \tilde{p}_l(g) \ln \frac{\tilde{p}_l(g)}{\tilde{p}_l^{(1)}(g)} + \sum_l \tilde{p}(l) \sum_g \tilde{p}_l(g) \ln \frac{\tilde{p}_l^{(1)}(g)}{\tilde{q}_l(g)} + D\big(\tilde{p}(l) : \tilde{q}(l)\big) \\
&= \mathrm{E}_l \left\{ I(\tilde{p}_l) \right\} + \mathrm{E}_l \left\{ D\big(\tilde{p}_l^{(1)} : \tilde{q}_l\big) \right\} + D\big(\tilde{p}(l) : \tilde{q}(l)\big) \ , \qquad (5)
\end{aligned}
$$

where we introduced $\mathrm{E}_l \left\{\cdot\right\}$ as the expectation over $\tilde{p}(l)$ (which depends on ϕ). The entropy of p can be written as

$$
\begin{aligned}
H(p) &= -\sum_g p(g) \ln p(g) = -\sum_l \tilde{p}(l) \sum_g \tilde{p}_l(g) \ln[\tilde{p}_l(g)\, \tilde{p}(l)] \\
&= -\sum_l \tilde{p}(l) \Big[\sum_g \tilde{p}_l(g) \ln \tilde{p}_l(g) + \sum_g \tilde{p}_l(g) \ln \tilde{p}(l) \Big] \\
&= \sum_l \tilde{p}(l) H(\tilde{p}_l(g)) + H(\tilde{p}(l)) \\
&= \sum_l \tilde{p}(l) \Big[\sum_{i=1}^{l} H(\tilde{p}_l^i) - I(\tilde{p}_l) \Big] + H(\tilde{p}(l)) \\
&= \mathrm{E}_l \left\{ \sum_{i=1}^{l} H(\tilde{p}_l^i) \right\} - \mathrm{E}_l \left\{ I(\tilde{p}_l) \right\} + H(\tilde{p}(l)) \ . \qquad (6)
\end{aligned}
$$

Adding equations (5) and (6) we find

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

$$
\begin{aligned}
D\big(p :q\big) &= \mathrm{E}_l \left\{ I(\tilde{p}_l) \right\} + \mathrm{E}_l \left\{ D\big(\tilde{p}_l^{(1)} : \tilde{q}_l\big) \right\} + D\big(\tilde{p}(l) : \tilde{q}(l)\big) \qquad (7) \\
&= \mathrm{E}_l \left\{ \sum_{i=1}^{l} H(\tilde{p}_l^i) \right\} + \mathrm{E}_l \left\{ D\big(\tilde{p}_l^{(1)} : \tilde{q}_l\big) \right\} + D\big(\tilde{p}(l) : \tilde{q}(l)\big) + H(\tilde{p}(l)) - H(p) \ . \qquad (8)
\end{aligned}
$$

The second RHS term in equation (8) 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 equation (8) 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:

$$\mathrm{E}_l\left\{\sum_{i=1}^{l} H(\tilde{p}_l^i)\right\} \le \log|\mathcal{A}|\ \mathrm{E}_l\left\{\sum_{i=1}^{l} 1\right\} = \log|\mathcal{A}|\ \mathrm{E}_l\left\{l\right\} = L_p \log|\mathcal{A}|\ , \qquad (9)$$

where we introduce $L_p = \mathrm{E}_l\{l\}$ as the expected description length of samples of p in the encoding ϕ. On the other hand, from (6), we get

$$\mathrm{E}_l\left\{\sum_{i=1}^{l} H(\tilde{p}_l^i)\right\} \ge H(p) - H(\tilde{p}(l))\ . \qquad (10)$$

Note that for an optimally compact coding $L_p \log|\mathcal{A}| = H(p) - H(\tilde{p}(l))$,[4] and both bounds are exact.

We collect these findings in

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

$$0 \le D\big(p:q\big) - \mathrm{E}_l\left\{D\big(\tilde{p}_l^{(1)} : \tilde{q}_l\big)\right\} - D\big(\tilde{p}(l) : \tilde{q}(l)\big) \le L_p \log|\mathcal{A}| - H(p) + H(\tilde{p}(l))\ .$$

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

$$D\big(p:q\big) = \mathrm{E}_l\left\{D\big(\tilde{p}_l^{(1)} : \tilde{q}_l\big)\right\} + D\big(\tilde{p}(l) : \tilde{q}(l)\big)\ ,$$

and $D\big(p:q\big)$ can easily be minimized by adapting the genotype marginals $\tilde{q}_l^i$ and the length distribution $\tilde{q}(l)$ (which can be set 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 (7) and (8) of lemma 1. We related the term $\mathrm{E}_l\left\{\sum_i H(\tilde{p}_l^i)\right\}$ to the description L_p via the bound (9) 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 $\mathrm{E}_l\left\{\sum_i H(\tilde{p}_l^i)\right\}$; it can equally be minimized by reducing the marginal entropies $H(\tilde{p}_l^i)$, which means not to exploit the expressional power of the alphabet. This can better be understood going back to expression (7) involving the mutual information: Ultimately, what matters is to reduce $\mathrm{E}_l\{I(\tilde{p}_l)\}$, i.e. finding a factorial code, which can also be done perfectly with very low marginal

[4] 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 exact bound is $L_p \log|\mathcal{A}| = H(p)$, as for the Shannon-Fano code.

entropies, not exploiting the alphabet. By relating it to the description length we showed that a compression is reducing $\mathrm{E}_l \left\{I(\tilde{p}_l)\right\}$ while *additionally* trying to exploit the alphabet optimally. Thus, compression is doing “more than we need” from the strict point of view of minimizing $D\big(p:q\big)$. Clearly, this also means that an optimally compact coding is not the only solution to minimize $D\big(p:q\big)$ 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 be set equal to $\tilde{p}_l^{(1)}$ and $\tilde{p}(l)$, thus perfectly minimizing $\mathrm{E}_l \left\{D\big(\tilde{p}_l^{(1)}:\tilde{q}_l\big)\right\}+ D\big(\tilde{p}(l):\tilde{q}(l)\big)$. In this case, we have

$$D\big(p:q\big) = \mathrm{E}_l \left\{I(\tilde{p}_l)\right\} = \mathrm{E}_l \left\{\sum_{i=1}^{l} H(\tilde{p}_l^i)\right\} + H(\tilde{p}(l)) - H(p) \ ,$$

and the problem reduces to finding an encoding that extinguishes the mutual information $\mathrm{E}_l \left\{I(\tilde{p}_l)\right\}$ or, as discussed, an optimal compression.

3. The two step procedure: If $\tilde{Q}$ is additionally constrained by a bound on the entropy, minimizing $D\big(p:q\big)$ remains a coupled problem of reducing $\mathrm{E}_l \left\{I(\tilde{p}_l)\right\}$ by a proper choice of ϕ and reducing $\mathrm{E}_l \left\{D\big(\tilde{p}_l^{(1)}:\tilde{q}_l\big)\right\}+D\big(\tilde{p}(l):\tilde{q}(l)\big)$ 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 $D\big(p:q\big)$ by an amount $\le L_p\, \hbar$, when $\hbar$ is the lower bound on the entropy in each marginal $\tilde{q}_l^i$. Note that a compact coding minimizes this worst case error.

5 Compression EAs

This section aims to provide a perspective on how the ideas on compact codes motivate the design of new Evolutionary Algorithms. We will propose a general scheme to design such algorithms but can present preliminary results only for a special case that is more in the line of conventional GAs rather than EDAs and close to algorithms proposed earlier. The results are promising. Most importantly though, the experiments exhibit what are crucial aspects to be discussed when the aim are practical implementations of the principle of compact codes.

A straight-forward way to design a new Evolutionary Algorithm, exploiting the idea of compact codes, is to combine any compression technique with any standard EA. Such a *Compression EA* reads

1. Initialize a finite population $p = \{p_1, .., p_\mu\}$.
2. Use a compression technique to find an encoding ϕ that (approximately) minimizes $L_p = \sum_{i=1}^{\mu} \mathsf{length}(\phi^{-1}(p_i))$.

3. Apply standard operators (e.g., mixing or EDA-operators) to generate λ offspring genotypes from the μ compressed parent genotypes.

4. Map the λ offspring genotypes from G back to P using ϕ.

5. Apply evaluation and selection on these offspring to generate the new parent population p' and repeat from step **2**.

The operators in step **3** may be any standard operators used in Evolutionary Algorithms. They have to be memory-less though, since the encoding will change in each iteration step and thus integrating knowledge from previous time steps becomes futile (cf. the discussion of cumulative compression as an alternative in section 6).

For instance, a most direct implementation of KL-search calculates the length and marginal distributions before resampling them (very similar to the PBIL algorithm, but accounting for variable size strings). This leads to the following *Compression EDA*:

3.a Calculate the length distributions $\tilde{p}(l)$ and the symbol marginals $\tilde{p}^i_l$ for each l from the compressed population $\tilde{p}$.

3.b Set $\tilde{q}(l) = \tilde{p}(l)$ and add entropy by mixing the marginals with the uniform symbol distribution $\mathcal{U}$, $\tilde{q}^i_l = (1-\alpha)\,\tilde{p}^i_l + \alpha\,\mathcal{U}$.

3.c Take λ samples from the distribution $\tilde{q}$ by recursively (i) picking an l from $\tilde{q}(l)$, (ii) $\forall^l_{i=1}$ pick a symbol g^i from $\tilde{q}^i_l$, and (iii) store $g = (g^1, .., g^l)$ as a new offspring genotype.

We must leave it to future work to present results on this Compression EDA. In the following we will investigate a concrete algorithm that is based on conventional mutation operators rather than such an EDA and similar to algorithms proposed earlier [7, 20]. The compression technique is based on L-systems, very simple, but computationally expensive. It is inspired by the work of Nevill-Manning and Witten [13]. Below we will briefly report on some experiences when using Lempel-Ziv compression (i.e., the algorithm of GZIP) instead.

5.1 A L-system Compression GA

The Compression. Step **2** of a Compression EA requires to find an encoding ϕ that minimizes the description length of individuals averaged over the current parent population. A simple technique of compression is to recursively analyze the samples for frequent pairs of neighboring symbols and replace such pairs by new symbols (cf. [13]). We realize this scheme with an L-system:

A L-system is a sequence $\Pi = \langle \pi_1, .., \pi_k \rangle$ of k productions π_i. Given the alphabet $\mathcal{A}$, each production $\pi = \langle l : r_1, .., r_m \rangle$ consists of a LHS symbol $l \in \mathcal{A}$ and a sequence $\langle r_1, .., r_m \rangle$ of RHS symbols. The L-system Π defines a mapping ϕ from one sequence s to another by applying all productions π_i, in the given order, on s. Applying a production $\langle l : r_1, .., r_m \rangle$ on s means to replace every l that occurs in s by the sequence $\langle r_1, .., r_m \rangle$. For instance, if the L-system is $\Pi = \langle \mathsf{e{:}cd}, \mathsf{f{:}dc}, \mathsf{c{:}ab}, \mathsf{d{:}ba} \rangle$, then a sequence $\langle \mathsf{ef} \rangle$ is mapped to $\phi(\langle \mathsf{ef} \rangle) = \langle \mathsf{abbabaab} \rangle$. The mapping ϕ can easily be inverted by applying all productions, in reverse order, inversely on a sequence. Inverse application of a production $\langle l\!:\!r_1,..,r_m\rangle^{-1}$ on s means to replace every subsequence $\langle r_1,..,r_m\rangle$ that occurs in s by l.

Let $\mathcal{A}$ be the non-negative integer numbers, $\mathcal{A} = \mathbb{N}_0$. Starting with a population $p = \{s_1,..,s_\mu\}$ of $\{0,1\}$-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. This scheme reads

2.a Initialize the L-system as $\Pi = \langle\,\rangle$ and the *new*-symbol as $c = 2$.

2.b Calculate the frequency of every symbol pair that occurs in the population.

2.c For every pair $\langle r_1, r_2\rangle$ of symbols that occurs more often than once, create a new production $\langle c\!:\!r_1, r_2\rangle$, append it to the *beginning* of Π, and increment the *new*-symbol $c \leftarrow c+1$. This is to be done in order, beginning with the pair of highest frequencies, and random order between pairs of same frequency. If there is no such pairs, exit the recursion.

2.d Recode the population $p \leftarrow \{\phi^{-1}(p_1),..,\phi^{-1}(p_\mu)\}$ (effectively, only the new productions will result in replacements).

2.f Repeat from step **2.b**.

The entropy. Following the general scheme of a Compression EA, we next need to specify a mixing operator in step **3**. This operator is supposed to induce the necessary entropy in $\tilde{q}$. We discussed earlier that this entropy should exceed the entropy $H(p)$ of the parent population to ensure further exploration, and we handled this issue theoretically by putting a lower bound on the entropy of feasible search distributions in the minimization of the Kullback-Leibler divergence.

However, first experiments showed quickly that there is an interesting problem when using the above compression in a straight-forward manner. Actually, the compression can be considered as too good and eventually violating the constraint of the lower bound on entropy: In the case of a *finite population*, the compression scheme often leads to what we term a *full compression*, where in the end every individual is represented by a different *single* symbol. The compressed population thus is only a set of μ different single symbols, and the description length was minimized down to 1. The mutual information indeed vanishes for the full compression. However, on this representation it is impossible to induce more entropy than the parent population had (which is, if they are disjoint, $\log \mu$). The parent population already has maximal entropy under all distributions over only one symbol (since $\tilde{q}(l) = \tilde{p}(l)$ is fixed) of the alphabet $\{1,..,\mu\}$. Thus, the full compression violates the constraint of the lower bound on entropy of $\tilde{q}$ and is thus infeasible.

A solution to this problem with the above compression scheme would be to stop compression at some level, maybe at the price that the mutual information is not fully extinguished, but allowing for the addition of entropy. We follow this approach by choosing the level of compression stochastically or, equivalently, to add entropy on different levels of compression.

Concretely, the algorithm stochastically decides whether to apply one-point mutations on each level of compression, i.e., in each recursion of the compression scheme. We insert the following step in the compression recursion:

2.e For each individual, decide with a probability α whether to apply a one-point mutation. A one-point mutation randomly picks a location and, with equal probabilities, deletes it, replaces it with a new symbol in $\{0, .., c-1\}$, or inserts such a new symbol.

This competes the description of the algorithm which omits an additional step **3**.

The Hierarchical XOR problem. The fitness function we consider is the Hierarchical XOR (HXOR) function [26, 7]. For a string $s \in \{0,1\}^n$ we first define a boolean function $h(s) \in \{0,1\}$, determining whether s is "valid" or not: Let $n = \text{length}(s)$ be the string length, $l \in \mathbb{N}_0$ such that $n/2 \leq 2^l < n$ ($l = \lfloor \log_2(n-1) \rfloor$), and $L = s_{1:2^l}$ and $R = s_{2^l+1:n}$ the left and right parts of the string when cut at location 2^l. Then, for $n \geq 2$, we define

$$h(s) = 1 \iff \Big[n = 2^{l+1} \ \wedge \ h(L) = 1 \ \wedge \ h(R) = 1 \ \wedge \ L = \bar{R} \Big] \ ,$$

and $h(s) = 1$ if $n = 1$. Here $\bar{R}$ is the bit-wise negation of the right part. The last condition means that the bit-wise XOR between left and right part must be true for each bit. For instance, the strings for which $h(s) = 1$ 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 $H(s) \in \mathbb{N}$, for $n \geq 2$,

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

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

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

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

Parameters. For the experiments, we use simple (μ,λ)-selection with population sizes $\mu = 30$ and $\lambda = 100$ and one elitist. The mutation probability (as indicated in step **2.e** of the algorithm) is $\alpha = 0.1$. The population was initialized with all $\langle 01 \rangle$ individuals.

Results. We tested the algorithm on the HXOR problem. Figure 1A displays the fitness trajectories for 20 runs on the 1024-bit HXOR problem. All runs consistently found the optimal 1024-bit string. It took on average 519 generations to reach this global optimum (with standard deviation ±168 generations).

Figure 1B displays the average first hitting generation and standard deviation for different length HXOR problems, up to a problem size of 8192 bits. (Beyond

Table 1. A L-system found to compress a population of identical solutions to the 1024-bit HXOR. 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 and, for brevity, only displayed the first 15 productions.

C:10
D:110
E:0110
F:1 0110
G:01 0110
H:01 1001 0110
I:1001 1001 0110
J:1001 0110

K:0110 1001 1001 0110
L:0110 1001 0110
M:0110 1001 0110 1001 1001 0110
N:1001 0110 0110 1001 1001 0110
O:1001 0110 0110 1001 1001 0110 0110 1001 0110 1001 1001 0110
P:1001 0110 0110 1001 0110 1001 1001 0110
etc...

that, exhaustive memory swapping slowed the algorithm too much down.) The algorithm found the optimum in all runs for all problem sizes and the variance of the first hitting generation is relatively small for a stochastic search scheme. The diagram also shows that the number of generations needed to find an optimum seems to grow linearly with the problem size.

To give an impression on the codings developed by the L-system compression scheme, table 1 displays an L-system found to compress a population of identical solutions to the 1024-bit HXOR problem. As discussed above, this compression implements a *full compression* such that eventually every individual 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 with productions are added to the L-system is stochastic (cf. step **2.c**). Consequently, the coding is not strictly hierarchical as a human might have designed it (or DevRep, see below) and the L-system comprises more pair-productions than minimally necessary to encode the sequence of length 1024 (which is 19).

Generally, the performance of the algorithm—in terms of the generations needed—is of the same order as the DevRep algorithm presented by de Jong [7], which is the only previous algorithm we are aware of capable of solving large HXOR problems. ([7] reported only on results for the 64-bit and 1024-bit HXOR problem, where about $2.3{\cdot}10^7$ 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 perform extremely bad on the HXOR problem because of its complex deceptiveness when search is performed on an atomic representation (see [7] for experiments).

Notes on GZIP. The compression scheme used is computationally very expensive. We argue below that a proper solution might be a cumulative approach to find compressions which are not recalculated in each generation. Alternatively, one

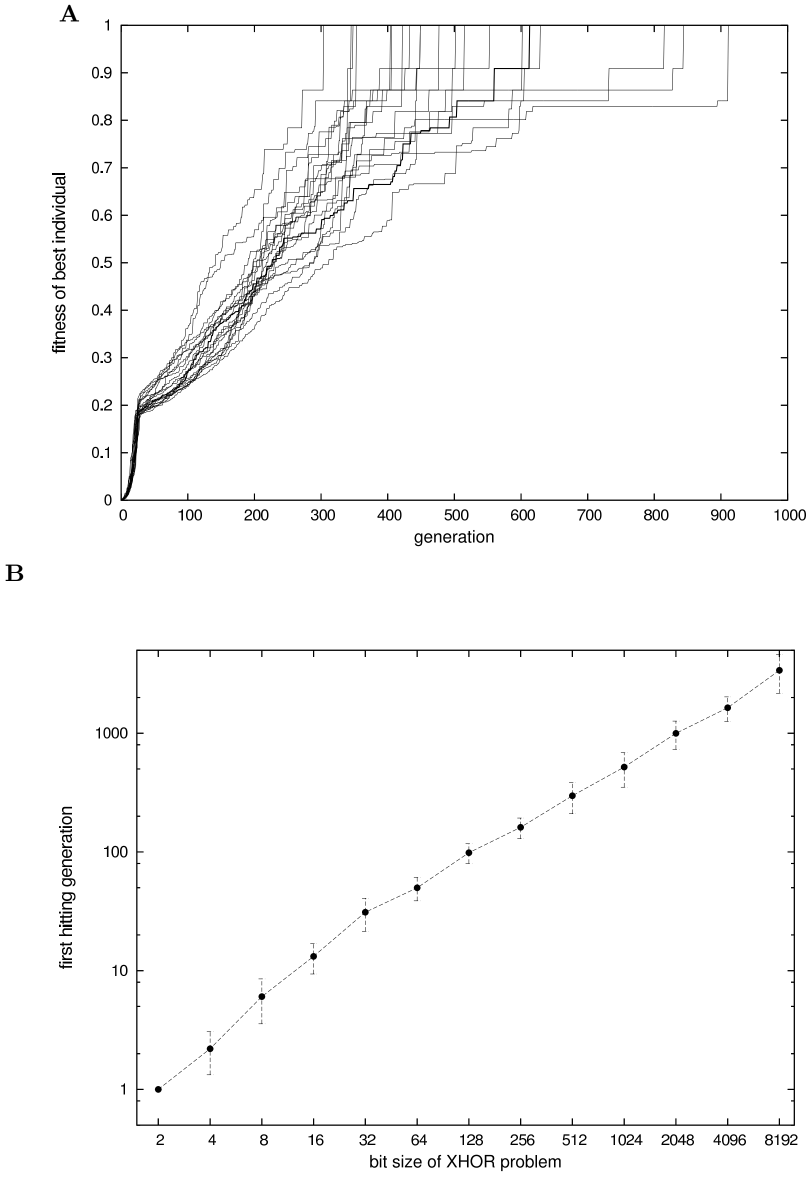

Fig. 1. A. The trajectory of the best individual fitness for 20 runs on the 1024-bit HXOR problem. One of the runs is drawn bold. **B.** First hitting generations, averaged over 20 runs for each of the different HXOR problem sizes.

Table 2. The first line is the original string, composed of 15×`abcd`. The lines below show 20 variations, generated by compressing the string with LZ-compression, applying a single one-point mutation, and decompressing again. Dots indicate that the symbol has not varied compared to the original string

```
abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcd
.a...a...a...a...a...a...a...a...a...a...a...a...a...a...a..
..........................................d.ddabcdabcd
..............................................abdabcdabcd
........ddabcdabcddbcdd.dddbcddbcddbcddbcddbcdabcdabcd
.....................cabcdabcdac.caccdaccdaccdabcdabcd
......aabadabcdabaabcdab.dab.dabcdabcdab.aba..a...a...a.
........bcdabcdabcdbbcdbcdbcd.bcd.bcd.bcd.bcd.bcdabcdabcd
d...d...d...d...d...d...d...d...d...d...d...d...d...d...d...
......ab..a.......ab......a...a...........ab..ab..a...a...a.
............................................................
.................................cdacdaccdabcdabcdabcdabcd
......bc..b.......bc......b...b...........bc..bc..b...b...b.
........................cc.bcd.bcd.bcdabcdabcdabcdabcd
.............abcddaacdaabcd..........abcddaabcd..a...a...a..aacdaacd
...a...a...a...a...a...a...a...a...a...a...a...a...a...a...a
....dcd.cddbcddbcddbcddcdd..d...d...d...d...d...d...d...
...................................................bcdabcdab
.....................dabcabcdabcdadabca....adcdadcdadcdabcdabcd
............................................................
..................bcd.....................bcd.bcd.b...b...b.
```

might want to use efficient standard compression techniques, for example the Lempel-Ziv compression as used by GZIP. We performed some experiments also with this compression algorithm.

However, LZ-compression has some instructive practical drawbacks. Let us first consider to compress a single sequence, mutate it, and decompress it. The symbol mutations of the compressed string have to obey some constraints such that the compressed sequence remains a valid string that can be decompressed by the LZ algorithm (the possible symbol range at each location is different). Table 2 displays the result of 20 different one-point mutations on the compressed representation. Here, a second issue about LZ-compression becomes apparent. The mutations at the beginning of the string are less likely to be severe or modular than at the end of the string. The reason is that LZ-compression is a 1-pass scheme which builds up codes while parsing the string. In the beginning, no codes have been build up yet and the string remains largely uncompressed. This a priori asymmetry is very undesirable in the context of evolutionary exploration. A possible trick is to concatenate the same string several times, compress the concatenation, mutate only the part which represents the last copy of the string, and decompress.

When implementing an algorithm, we exploited this observation. To compress and mutate a single individual, we concatenated all individuals of the population and additionally appended the individual of interest into a long sequence and compressed it. In this manner, LZ-compression first uses the whole population sequence to build up codes that are then used to compress the last individual. Then we mutated the compressed representation of the last individual and decompressed it. A computationally very expensive scheme, again.

Experiments with the HXOR problem showed a very high variance in performance between runs. For the 128-bit HXOR problem it frequently occurred that runs never find the optimum while others find it within a few hundred generations. Generally, the experiments with LZ-compression gave some important insight in the relevance of how the codes are build up. They do however not support the practical use of conventional LZ-compression for evolutionary algorithms.

6 Discussion

Summary. The basis on which we developed the theoretical analysis was the indirect Estimation-of-Distribution scenario, where a distribution over P is estimated via a distribution over a genotype space G and an encoding $\phi: G \to P$. The genotype distribution is restricted to factorize. Thus the problem of estimating a distribution is split into "decoding" the structural aspects of the distribution (via compression) and then estimating the remaining structure-less factorized distribution. The main result are Lemmas 1 and 2 relating the KLD subject to minimization to the description length of the encoding.

The class of Compression EAs proposed in the last section are straightforward combinations of compression techniques with standard EA operators. An algorithm based on a simple type of L-system compression performs reliably well on the hierarchically deceptive HXOR function, which we tested up to a problem size of 8192 bits. The number of generations needed seems to grow linearly with the problem size.

However, the main aim of giving this explicit example for a Compression EA was to introduce to the following discussion of two aspects that seem to be important (i) when thinking about future, computationally efficient Compression EAs and (ii) when considering the implications of the presented theory for the understanding of natural evolution and the self-adaptive evolution of genetic representations.

Cumulative compression. The L-system compression we used has properties that proved beneficial in the experiments: unlike LZ-compression it is unbiased w.r.t. which modules are encapsulated and it can develop arbitrary hierarchies. To recompute the compression from scratch at every generation also has advantages: the encoding is always adapted to the current population, i.e. the currently available information about the problem, and the stochasticity of the compression scheme leads to more diversity in exploration at different generations.

Clearly though, recomputing the compression at every generation is computationally very expensive. The general scheme of a Compression EA should thus be modified to develop the compact encoding in a cumulative manner rather than recomputing it at every generation. In the case of L-system compression, cumulative could mean that the L-system is persistent over the generations and at each generation only a single new production is added or an old, unused production deleted. Assuming that the inherent structure of found solutions will not evolve too fast, the cumulative scheme would still allow to provide a good compression of the current population.

Besides being computationally much more effective, the cumulative approach has another interesting perspective. Since the encoding is incrementally build up during evolution, the encoding becomes a variable that integrates information about the successful solutions over more than one generation. This is comparable to strategy parameters in Evolution Strategies which, for instance, integrate the average movement of the population over the recent history, assuming that this search direction will also be profitable in the future [9]. Such schemes have hardly been transferred to ordinary GAs because the notion of "proceeding in the same direction" makes no sense on the hypercube. The notion does make sense though on the more abstract level of "proceeding by incorporating the same structural dependencies that have previously been successful"—where the notion of "search direction" is replaced by the notion of "structural properties of search". The encoding becomes the variable which is capable to integrate such information. The objective of compactness indicates how an adequate permanent adaptation can be achieved.

Self-adaptive σ-evolution of compact representations. In this work we addressed a *bijective* encoding ϕ that is adapted explicitly (externally) at each generation. This approach is complementary to the formalism we proposed earlier to describe the *self*-adaptive evolution of genetic representations [22, 21]. The proper way to formalize the self-adaptive case is to consider a *fixed* but *non-injective* genotype-phenotype mapping. In that case there exists a variety (neutral set) of different genotypes that map to the same phenotype. The "choice of genetic representation" here means which genotype from the neutral set is chosen to encode the phenotype. The specific example investigated in [20] clarifies the relation between the two complementary frameworks: A genotype in the self-adaptive scenario may be the *tuple* (g_0, Π) of a (compact) string g_0 (termed axiom, or egg cell) and an L-system Π (termed genome). The global genotype-phenotype mapping from the space of such tuples to phenotype is clearly fixed. But different genotypes, involving different L-systems Π, may map to the same phenotype and thus induce structurally completely different phenotypic variabilities. In this scenario, the evolution of genetic representations can be understood as moves in the neutral set, e.g., neutral reorganizations of the L-system Π.

In [22] it was show that the self-adaptive evolution of genetic representations is driven by an implicit selection which discriminates phenotypically equivalent genotypes by the quality of the phenotypic variability that they induce (which is related to the effective fitness [14, 19]). More precisely, the selectional advantage of different genetic representations of the same phenotype is proportional to the negative *Kullback-Leibler divergence* between the phenotypic variability and the Boltzmann fitness distribution (assuming a lower bound on phenotypic entropy, see [22] for details).

The objective of minimizing the Kullback-Leibler divergence is thus the key to transfer the results we derived here to the self-adaptive scenario, now stating that there is an effective selection pressure on the description length of a genetic representation. Indeed, a tendency towards compact representations was observed in experiments on σ-evolution [20]. It was previously argued that its origin is the advantage of mutational robustness when every genetic symbol underlies a constant mutation rate. We can now add another origin, namely that compact representations are *structurally* more favorable—meaning that the phenotypic variability they induce allows to reduce the Kullback-Leibler divergence and follow the Estimation-of-Distribution principle. However, it is yet open to which degree both effects contributed to the evolution of compact codes.

Acknowledgment

I would like to thank 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 Transactions on Information Theory*, 47(5):1701–1711, 2001.
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, Comp. Sci. Dep., Carnegie Mellon U., 1994.
3. S. Baluja and S. Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proc. of Fourteenth Int. Conf. on Machine Learning (ICML 1997)*, pages 30–38, 1997.
4. L. Barbulescu, J.-P. Watson, and D. Whitley. Dynamic representations and escaping local optima: Improving genetic algorithms and local search. In *Seventeenth National Conference on Artificial Intelligence (AAAI)*, pages 879–884, 2000.
5. A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44:2743–2760, 1998.
6. J. S. de Bonet, C. L. Isbell, Jr., and P. Viola. MIMIC: Finding optima by estimating probability densities. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 424. The MIT Press, 1997.
7. E. D. de Jong. Representation development from Pareto-Coevolution. In *2003 Genetic and Evolutionary Computation Conference (GECCO 2003)*, 2003.
8. G. Halder, P. Callaerts, and W. Gehring. Induction of ectopic eyes by targeted expression of the eyeless gene in Drosophila. *Science*, 267:1788–1792, 1995.
9. N. Hansen and A. Ostermeier. Completely derandomized self-adaption in evolutionary strategies. *Evolutionary Computation*, 9:159–195, 2001.
10. R. B. Heckendorn and A. H. Wright. Efficient linkage discovery by limited probing. *Evolutionary Computation*, 2004. Accepted for publication.
11. G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proc. of 2001 Congress on Evolutionary Computation (CEC 2001)*, pages 600–607. IEEE Press, 2001.
12. G. E. Liepins and M. D. Vose. Representation issues in Genetic Algorithms. *Journal of Experimental and Theoretical Artificial Intelligence*, 2, 1990.

13. C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.
14. P. Nordin and W. Banzhaf. Complexity compression and evolution. In L. Eshelman, editor, *Genetic Algorithms: Proc. of Sixth International Conf. (ICGA 1995)*, pages 310–317. Morgan Kaufmann, Pittsburgh, 15-19 1995.
15. M. Pelikan and D. E. Goldberg. Hierarchical BOA solves Ising spin glasses and MAXSAT. In *Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pages pp. 1271–1282. Springer-Verlag, 2003.
16. M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary Computation*, 9:311–340, 2000.
17. M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. Technical Report IlliGAL-99018, Illinois Genetic Algorithms Laboratory, 1999.
18. F. Rothlauf and D. E. Goldberg. Redundant representations in Evolutionary Computation. *Evolutionary Computation*, 11:381–415, 2003.
19. C. R. Stephens and J. M. Vargas. Effective fitness as an alternative paradigm for evolutionary computation I: General formalism. *Genetic Programming and Evolvable Machines*, 1:363–378, 2000.
20. M. Toussaint. Demonstrating the evolution of complex genetic representations: An evolution of artificial plants. In *2003 Genetic and Evolutionary Computation Conference (GECCO 2003)*, pages 86–97, 2003.
21. M. Toussaint. The evolution of genetic representations and modular neural adaptation, April 2003. PhD thesis, Institut für Neuroinformatik, Ruhr-Universiät-Bochum, Germany. Published with the Logos Verlag Berlin (2004), ISBN 3-8325-0579-2, 173 pages.
22. M. Toussaint. On the evolution of phenotypic exploration distributions. In C. Cotta, K. De Jong, R. Poli, and J. Rowe, editors, *Foundations of Genetic Algorithms 7 (FOGA VII)*, pages 169–182. Morgan Kaufmann, 2003.
23. M. Toussaint. Notes on information geometry and evolutionary processes, 2004. Los Alamos pre-print nlin.AO/0408040.
24. P. M. B. Vitányi and M. Li. Minimum Description Length induction, Bayesianism, and Kolmogorov complexity. *IEEE Trans. Inform. Theory*, IT-46:446–464, 2000.
25. G. P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Evolution*, 50:967–976, 1996.
26. R. A. Watson and J. B. Pollack. Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In *Late breaking papers at the Genetic and Evolutionary Computation Conference*, pages 292–297, 1999.
27. D. Whitley, S. Rana, and R. Heckendorn. Representation issues in neighborhood search and evolutionary algorithms. In *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*, pages 39–58. John Wiley Sons Ltd., 1997.
28. A. H. Wright, R. Poli, C. R. Stephens, W. B. Langdon, and S. Pulavarty. An Estimation of Distribution Algorithm based on maximum entropy. In *2004 Genetic and Evolutionary Computation Conference (GECCO 2004)*, pages 343–354. Springer, Berlin, 2004.