Inferring and Exploiting Problem Structure with Schema Grammar

Chris R. Cox and Richard A. Watson

Department of Electronics and Computer Science
University of Southampton, UK
{c.cox,r.a.watson}@soton.ac.uk

Abstract. In this work we introduce a model-building algorithm that is able to infer problem structure using generative grammar induction. We define a class of grammar that can represent the structure of a problem space as a hierarchy of multivariate patterns (schemata), and a compression algorithm that can infer an instance of the grammar from a collection of sample individuals. Unlike conventional sequential grammars the rules of the grammar define unordered set-membership productions and are therefore insensitive to gene ordering or physical linkage. We show that when grammars are inferred from populations of fit individuals on shuffled nearest-neighbour NK-landscape problems, there is a correlation between the compressibility of a population and the degree of inherent problem structure. We also demonstrate how the information captured by the grammatical model from a population can aid evolutionary search. By using the lexicon of schemata inferred into a grammar to facilitate variation, we show that a population is able to incrementally learn and then exploit its own structure to find fitter regions of the search space, and ultimately locate the global optimum.

Keywords: Generative grammar, compression, evolutionary algorithm, estimation of distribution algorithm, NK fitness landscape.

1 Introduction

The field of natural computing has proposed a variety of both implicit and explicit model-building algorithms that attempt to infer the structure of a problem from populations of above-average-fitness individuals. The intuition is that by using these models evolutionary search can be directed to promising areas of a high-dimensional fitness landscape by recombining multivariate features that are commonly found in fit individuals. The best known implicit model is variation by crossover in sexual genetic algorithms. The seminal work of Holland on "Schema Theorem" [1] and subsequently Goldberg on the "Building Block Hypothesis" [2] showed that, at least in theory, crossover offers a scheme under which fit multivariate patterns known as schemata can grow in frequency in an evolutionary population. Moreover, these schemata can act as genetic building blocks that can themselves be recombined, allowing evolutionary search to be performed

T. Bartz-Beielstein et al. (Eds.): PPSN XIII 2014, LNCS 8672, pp. 404–413, 2014.

at higher levels of organisation. More recently Estimation of Distribution Algorithms (EDAs) have been proposed that attempt to build explicit, probabilistic models of fit individuals that can be sampled in order to evolve a population through either constructive or perturbative variation (see [3] for a review). These methods have been shown to be competent at solving many different types of problem, including problems that have proven difficult for crossover-based GA's.

In this paper we present an alternative method of explicitly modelling problem structure. Unlike other explicit modelling techniques that build probabilistic models of a sample population using statistical inference, we build a *lossless* model using grammatical inference. By a lossless model we mean a model from which the original samples can be recreated without any loss of information. Indeed, we will show that the grammar induction algorithm used here is a type of lossless compression algorithm that identifies a hierarchy of genetic schemata with which the sample population can be more compactly described. A key feature of the "schema grammar" we introduce is that it generates combinatorial as opposed to sequential languages, with the rules it encodes producing unordered sets of symbols rather than ordered sequences or strings. This sets it apart from traditional sequential grammars and enables grammatical inference in combinatorial problem space for the first time.

We have recently demonstrated the value of this modelling technique in solving synthetic building block problems, showing that the simple modular structure in these problems can be correctly inferred and then reused to facilitate superior time complexity in global search [4]. In the present paper we investigate NK-landscape problems [5], which are irregular and unpredictable but contain an inherent statistical structure. We show that by information about this structure can be learned from a sample population of fit individuals using schema grammar, and that by using the the lexicon of schemata inferred into a grammar to facilitate variation, a population is able to locate fitter regions of the search space and ultimately locate the global optimum.

2 Compression Evolutionary Algorithms

Our objectives when building a model of fit individuals in a population are to identify any inherent structure within the fitness landscape, visible as variable dependencies, and exploit it within the evolutionary search process. Toussaint suggested that one way of achieving this is using Compression Evolutionary Algorithms [6]. By compressing a sample population of phenotypes any dependencies that are present in the population are factored into the structure of the compression model, and what is left is a compressed, decorrelated representation. The idea is that if we consider this compressed representation to be a genotype, and the compression model a genotype-phenotype map, then random perturbation of the genotype will produce phenotypes that obey the dependencies of the problem space. The concept is somewhat similar to the Grammatical Evolution (GE) techniques used in genetic programming [7] in which genetic programs are evolved using a grammar as a genotype-phenotype map, although in the GE

case the grammars are predefined using knowledge of the problem rather than inferred.

Toussaint was able to demonstrate the concept of Compression EA's on variable length problems using sequential grammar inference as the compression model, however the constraints of sequential grammars limit the practical usefulness of the technique. Specifically, only sequentially-contiguous variable dependencies can be modelled, and it is not possible to infer the length constraints or positional structure of a problem.[1] The approach we outline here is an type of Compression Evolutionary Algorithm, also using grammar inference as a compression model, however we overcome the limitations described above by using a set grammar that can produce combinatorial rather than sequential languages, and a genetic encoding with no intrinsic sequential order. This allows us to apply Compression Evolutionary Algorithms to a wider range of evolutionary problems.

3 Schema Grammar

Schema grammar is able to represent and compress any combinatorial expressions that can be encoded as an unordered set of *terminal symbols* which represent the "alphabet" of the problem space. In this paper we will consider n-dimensional binary spaces only. In order to encode genotypes and schemata in the required way we adopt the *Messy GA* encoding of Goldberg et al [9]. The encoding offers an alternative scheme to bit strings for addressing n-dimensional binary spaces using order-independent sets of $\langle locus|allele\rangle$ tuples, for example the bit string 0110 can be represented by the set $\{\langle 0|0\rangle, \langle 1|1\rangle, \langle 2|1\rangle, \langle 3|0\rangle\}$. In an n-dimensional problem the complete alphabet of terminal symbols, Σ, is just the set of all possible alleles:

$$\Sigma = \{\langle \lambda|\alpha\rangle : \lambda \in \{0, .., n\}, \alpha \in \{0, 1\}\} \tag{1}$$

The grammar is a type of context-free grammar (CFG) similar to the context-free grammar codes that are typically used for sequential compression (see [10] for a review). It has *straight-line* properties such that each *non-terminal symbol* (variable) in the grammar is only associated with one production rule and there are no loops in production. These properties ensure that productions are deterministic: in our context this ensures a surjective mapping from genotype to phenotype. Production rules in the grammar are expressed in terms of set membership relations, so using the previous example we can define a non-terminal symbol that represents an individual genotype g using the production rule:

$$g \rightarrow \{\langle 0|0\rangle, \langle 1|1\rangle, \langle 2|1\rangle, \langle 3|0\rangle\}$$

The encoding also provides a very natural way of representing variable interactions as genetic schemata, including those that may overlap in locus or allele

[1] The meta-Grammar Genetic Algorithm described in [8] manages to apply sequential grammars to solving concatenated trap problems by using "wrapping operators" and pre-defined GE grammars to overcome these limitations.

Table 1. An example grammar extract showing two genotypes that are specified using schemata. The phenotype expansion of each rule is shown in the second column using standard schema notation. Note that the example shown does not constitute a compact representation of the two genotypes - it is used to illustrate the nature of the G-P mapping.

$g_0 \rightarrow \{s_0, \langle 2\|0\rangle, \langle 3\|0\rangle\}$	$g_0 \stackrel{*}{\Rightarrow} 110011$
$g_1 \rightarrow \{s_3, \langle 0\|0\rangle, \langle 2\|1\rangle, \langle 3\|1\rangle, \langle 4\|0\rangle\}$	$g_1 \stackrel{*}{\Rightarrow} 011100$
$s_0 \rightarrow \{s_1, s_2\}$	$s_0 \stackrel{*}{\Rightarrow} 11{**}11$
$s_1 \rightarrow \{\langle 0\|1\rangle, \langle 5\|1\rangle\}$	$s_1 \stackrel{*}{\Rightarrow} 1{****}1$
$s_2 \rightarrow \{\langle 1\|1\rangle, \langle 4\|1\rangle\}$	$s_2 \stackrel{*}{\Rightarrow} *1{**}1*$
$s_3 \rightarrow \{\langle 1\|1\rangle, \langle 5\|0\rangle\}$	$s_3 \stackrel{*}{\Rightarrow} *1{***}0$

space. Moreover, the recursive properties of production rules allow schemata to be modelled using multiple levels of sub-schemata, which can be a far more compact representation if those sub-schemata are used in multiple places. These qualities are illustrated in Table 1 below.

We use each instance of schema grammar to represent a genetic population with the symbol table of the grammar providing a surjective mapping of genotypes G to phenotypes P. We specify the mapping as the recursive expansion of the production rules associated with each genotype (denoted using $\stackrel{*}{\Rightarrow}$):

$$P = \{p : g \stackrel{*}{\Rightarrow} p, g \in G\} \tag{2}$$

Similarly, we say that the phenotype mapping of each schema symbol in S is simply the recursive expansion of the production rules associated with each symbol. It is through this mapping mechanism that we later create genetic variation operators for evolutionary search. The phenotype mappings for both complete genotypes and individual schemata is shown in the second column of Table 1.

We can now formally define schema grammar as the 5-tuple:

$$\mathbb{G}_{Schema} = (V, G, S, \Sigma, R), \text{ where:} \tag{3}$$

- V is a set of non-terminal symbols (variables), each of which defines a sub-language of $\mathbb{G}_{Schema}$, where $V \ni \{G, S\}$ and $G \cap S \in \emptyset$
- G is a non-empty set of non-terminal symbols representing the genotypes in the sample population
- S is a set of non-terminal symbols representing genetic schemata, which may be empty
- Σ is the set of terminal symbols specified in equation (1)
- R is a finite set of production rules that relate each non-terminal symbol to the expansion of an unordered set of symbols, such that $R : V \rightarrow \{V \cup \Sigma\}^*$

4 Grammar Inference

In this section we describe an offline lossless compression algorithm that is able to infer an instance of schema grammar from a population of sample genotypes. The algorithm we use is based on an adaptation of two existing sequential compression algorithms that use grammar codes: *RE-PAIR* (recursive pairing), an offline algorithm from Larsson and Moffat [11] and *SEQUITUR*, an online algorithm from Nevill-Manning and Witten [12]. The central principle of both our algorithm and the sequential algorithms from which it derives is to infer dependency from frequency of co-occurrence. In sequential compression we define co-occurrence as two symbols appearing next to each other (in order) in a string. In schema grammar we define co-occurrence as two symbols being members of the same set, which is order independent. For example, in the sequential production rule $x \rightarrow abc$ the two co-occurrences present are ab and bc. In schema grammar the co-occurrences in the production rule $x \rightarrow \{a, b, c\}$ are the 2-combinations $\{a, b\}, \{a, c\}, \{b, c\}$.

Compression starts by creating a grammar with no schemata ($S = \emptyset$) and a direct mapping of genotypes to phenotypes using the Messy-GA encoding. The process proceeds iteratively in a way analogous to $RE-PAIR$: on each iteration the two most frequently co-occurring symbols in the genotype encodings G are identified, choosing randomly if two co-occurrences have the same frequency. A new non-terminal symbol is created that expands to the symbol pair and is added to S. All co-occurrences of the two symbols in the samples are then substituted with a single occurrence of the new non-terminal symbol. This process continues recursively until no two symbols co-occur anywhere in the grammar more than once. Larger schemata are formed by the recursive substitution of non-terminal symbols.

We also adopt the rule utility constraints of *SEQUITUR* which allow unnecessary nested hierarchies of symbol pairs to be collapsed into larger symbols. The procedure is simply implemented by taking any non-terminal symbols that

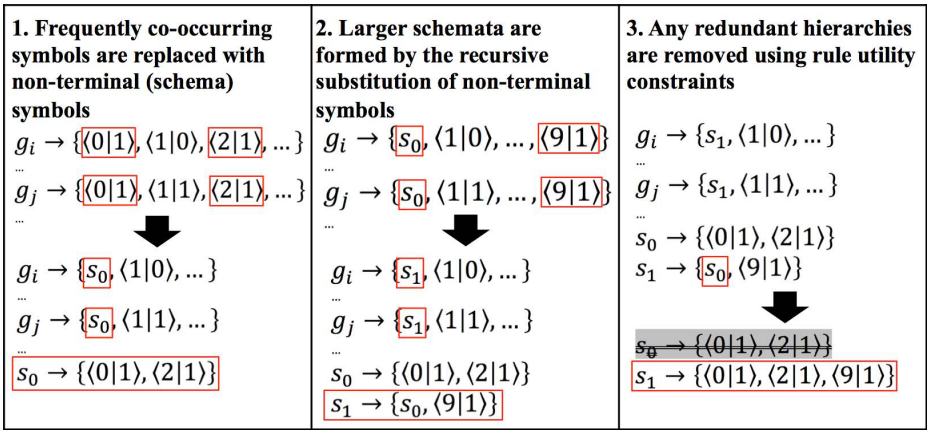

Fig. 1. Illustration of the offline compression algorithm

are referenced only once in the production rules and reversing the symbol substitution. This situation can occur when a particular combinatorial pattern of size $|s| > 2$ only ever appears whole in the samples, with subsets of the pattern not appearing more frequently. Rather than creating a nested hierarchy of $|s| - 1$ non-terminal symbols only one is required. The end result of the process is a compressed, decorrelated genotype representation and a G-P map implemented using a hierarchy of schemata. The operation of the algorithm is illustrated in Fig 1.

5 Inferring and Exploiting Problem Structure

In this section we investigate the ability of the grammar to infer problem structure from fit individuals on NK-landscapes [5], and then exploit the structure to improve evolutionary search. NK-landscape problems use a simple bit-string representation for candidate solutions and are parameterised by n, which specifies the problem size (in bits), and k, which specifies the number of "neighbours" of each bit. We use the nearest neighbour variant of the problem such that the neighbours of each bit overlap in an ordered way (see [5] for details), but shuffle the bits to remove any explicit sequential linkage. By varying neighbourhood size k we are able to tune the degree of correlation structure present in the problem from a completely correlated, unimodal landscape (where $k = 0$) to a completed uncorrelated, random landscape ($k = n - 1$).

Fitness is calculated as the sum of individual bit fitnesses, each of which is a function of its own state and that of the neighbours to which it is connected. A lookup table f_i is created for each bit that maps each of the 2^{k+1} possible input states associated with bit i and its neighbours to a random fitness value in the range 0 to 1. We define the fitness F of a n-bit bitstring X with state $(x_0, x_1, ..., x_{n-1})$ as:

$$F(X) = \frac{1}{n} \sum_{i=0}^{n-1} f_i(x_i, \Omega(i)) \tag{4}$$

Where $\Omega(i)$ is the state of the k neighbours of bit i. Within the framework of schema grammar a phenotype bitstring X is generated from a schema grammar genotype g by the phenotypic expansion of the production rules associated with g (see Section 3), and rendering the resulting set of terminal symbols as a bit string.

In order to identify individuals of above-average fitness we used the *schema search* process described in Algorithm 1 below. Schema search is a stochastic local search process that is able to use the symbols of schema grammar as variation operators. The search operates in phenotype space and implements phenotypic variation using the recursive expansions of each variation symbol (overwriting any symbols occupying the same loci). The set of symbols used for *schema search* specify the search neighbourhood, and when it is used with terminal symbols (i.e. individual allele values) it is equivalent to a stochastic bit-flip hill climber.

Algorithm 1. Schema Search Pseudocode

```
input  : initial phenotype bitstring p_i
input  : variation symbol set Λ
output: phenotype bitstring p_o
p_o ← p_i
for λ ∈ RandomPermutation(Λ) do
    candidate_p ← p_o
    ExpandInto(candidate_p, λ)
    if F(candidate_p) > F(p_o) then
        p_o ← candidate_p
    end
end
```

We generated populations of fit individuals on NK-landscapes with varying epistasis k. For each population we randomised phenotypes and use schema search with terminal symbol (bit-flip) variation operators, and we also ensured that each individual was unique in the population. The resulting phenotypes were compressed into an instance of schema grammar using the method detailed in Section 4. We then measured the compressibility of each population using the entropy of the resulting grammar code (as specified in [10]). This is illustrated in Figure 2, together with a control which is the entropy of a grammar induced from random bit strings of the same size. The figure shows that when there is minimal epistasis ($k = 1$), with a significant degree of correlation structure present in the landscape, the sample population is highly compressible with low entropy. As epistasis is increased then the compressibility of the population is reduced until, in the limit, it is no more compressible than a population of random bit strings.

These results suggest that grammar induction is detecting structure where it is present and using it to compress a population. However by themselves the results do not confirm whether it is detecting the "right" structure, or structure that can be exploited to improve evolutionary search. Some further insight can be gained by looking at the schemata inferred from the population. Figure 3 shows the inferred schema hierarchy for a single above-average fitness individual on an NK-landscape (shown unshuffled for presentation). Although the structure is broadly irregular, as may be expected given the randomised nature of the landscape, patterns reflecting the intrinsic neighbourhood structure of the problem can clearly be seen in the grammar, particularly at higher levels. This decomposition suggests that the schema structure in the grammar may be encoding useful neighbourhood structure from the problem landscape. The figure also illustrates the hierarchical nature of the grammar, with larger building blocks forming from multiple building blocks at lower levels.

We then used a multi-scale search algorithm to investigate whether the information contained in the compression model of a fit population could be exploited to aid evolutionary search. We created a population of fit individuals, each initialised using schema search (Algorithm 1) with bit flip variation operators, which were then modelled using schema grammar induction as described

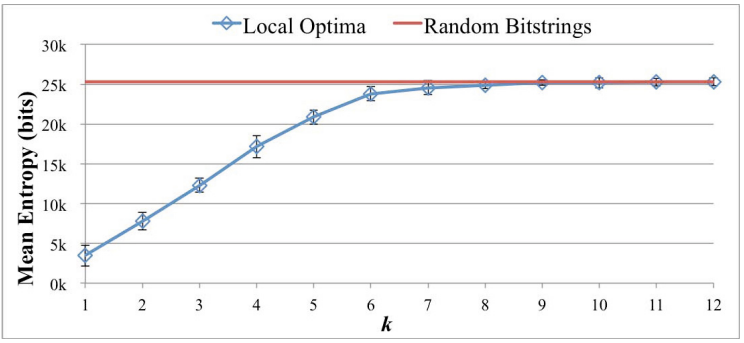

Fig. 2. The average entropy of compressed grammars for populations of local optima with increasing k ($n = 50$, population size $= 200$). As k increases the compressibility of a population reduces, indicating a reduction in detectable structure.

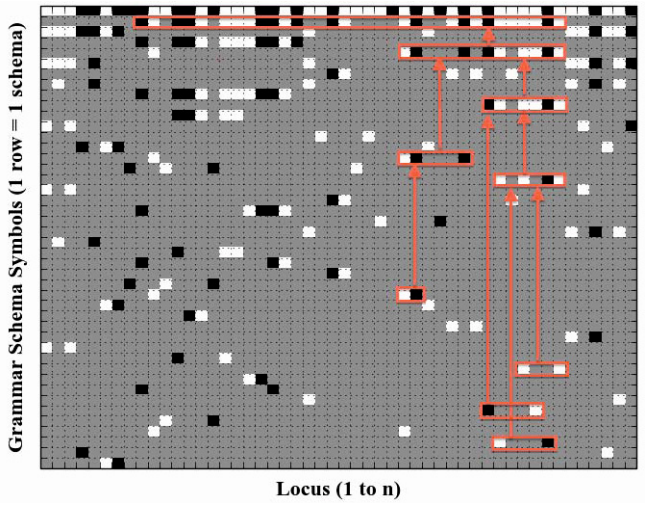

Fig. 3. An example schema grammar hierarchy for a single fit individual on an NK landscape ($n = 50, k = 5$, shown unshuffled). The top row shows the complete individual (black = 0, white = 1 at each locus), with the schemata it is compressed with shown in the rows below (in dependency order). The annotations illustrate part of the dependency hierarchy.

in previous sections. We examined the mutant spectra of the population using the schema symbols in the compressed representation of the population as variation operators (all symbols present in the production rules of each genotype). Although each mutation is a point mutation in compressed genotype space, the hierarchical grammatical expansions of the symbols in phenotype space include mutations with many interacting variables. We compared against two controls: random macro-mutation variation using an instance of the grammar but with shuffled terminal symbols, and uniform crossover and mutation (mutation rate $= 1/n$). The results in Figure 4 from a representative problem instance show

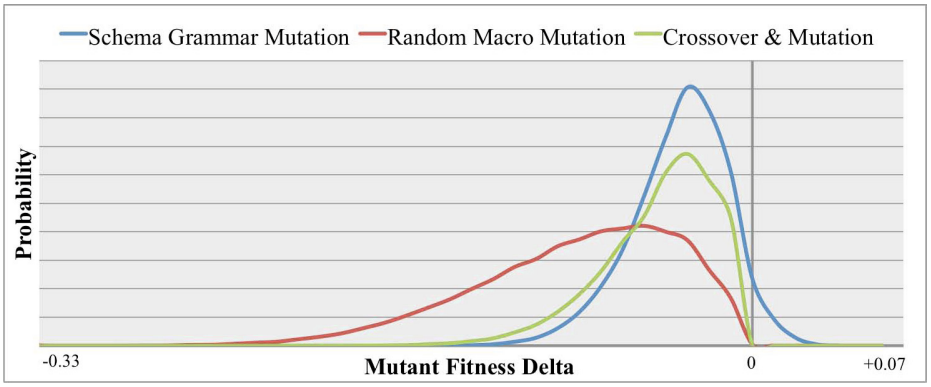

Fig. 4. Mutant spectra of a fit population using schema grammar to facilitate variation, as well as two controls ($n = 50$, $k = 4$, population size $= 200$). The area under the positive tail of each distribution represents fitness-improving mutations, which were 7.64% of schema grammar mutations, compared to 0.11% of random macro-variations and 0.04% of crossover/mutation variations. The average hamming distance of fitness-improving grammar mutations was 5.66 bits.

that inferred schema structure provides a variation neighbourhood that includes many more fitness-improving mutations than either of the two controls.

We conducted a second experiment to investigate the extent to which the fitness of a population could be improved, solely using the specific schema structure inferred by the grammar to facilitate macro-variation. On each iteration of the algorithm the population was modelled using schema grammar, then schema search was run on the population using the grammar's schema symbols as variation operators. If any beneficial moves were found then this process was repeated, continuing until more moves could be made or the global optimum was located. To help maintain diversity any duplicate phenotypes in the population were re-initialised prior to compression. We ran the algorithm on multiple, random NK-landscapes, with the global optima identified in advance using Pelikan's branch and bound solver [13]. We investigated problems between lengths $n = 20$ and $n = 50$ with k varying in the range $1 - 5$, and tested 400 random problem instances for each configuration. In every run of the algorithm the global optimum was successfully located. In less than 2% of problem instances it was found using single-bit hill-climbers (particularly when $k = 1$), however in all other cases the schema structure inferred from the population contained information that allowed search to continue and find fitter regions of the search space, until ultimately the global optimum was found. Further work is planned to test the scalability of these techniques, including comparisons with other model-building solvers such as BOA and LTGA, which are able to efficiently solve nearest-neighbour NK landscapes [14,15].

6 Conclusions

In this paper we have introduced a new class of generative grammar that is capable of modelling combinatorial structure. We have demonstrated that on NK-landscape problems, schema grammar is able to compress a population of individuals using a hierarchy of schema symbols that reflect the intrinsic structure of the landscape. We have also shown that the schemata inferred into the grammar can be exploited by facilitating multi-scale variation during evolutionary search. Our results suggest that the schema grammar is an effective type of compression EA that can demonstrably discover and exploit complex structural regularity in evolutionary problem solving.

References

1. Holland, J.H.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U. Michigan Press (1975)
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional (1989)
3. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation 1(3), 111–128 (2011)
4. Cox, C.R., Watson, R.A.: Solving Building Block Problems using Generative Grammar. In: Proceeding of the 2014 Conference on Genetic and Evolutionary Computation, pp. 341–348. ACM (2014)
5. Kauffman, S.A.: The Origins of Order. Self-organization and Selection in Evolution. Oxford University Press (1993)
6. Toussaint, M.: Compact representations as a search strategy: Compression EDAs. Theoretical Computer Science 361(1), 57–71 (2006)
7. Ryan, C., Collins, J.J., Neill, M.O.: Grammatical evolution: Evolving programs for an arbitrary language. Genetic Programming, 83–96 (1998)
8. O'Neill, M., Brabazon, A.: mGGA: The meta-grammar genetic algorithm. Genetic Programming, 311–320 (2005)
9. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems 3(5), 493–530 (1989)
10. Kieffer, J.C., Yang, E.: Grammar-based codes: a new class of universal lossless source codes. IEEE Transactions on Information Theory 46(3), 737–754 (2000)
11. Larsson, N.J., Moffat, A.: Off-line dictionary-based compression. Proceedings of the IEEE 88(11), 1722–1732 (2000)
12. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research 7, 67–82 (1997)
13. Pelikan, M.: Analysis of estimation of distribution algorithms and genetic algorithms on NK landscapes. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 1033–1040. ACM (March 2008)
14. Pelikan, M., Sastry, K., Goldberg, D.E., Butz, M.V., Hauschild, M.: Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 851–858 (2009)
15. Thierens, D.: The linkage tree genetic algorithm. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 264–273. Springer, Heidelberg (2010)