Towards Memoryless Model Building

David Iclănzan
david.iclanzan@gmail.com

D. Dumitrescu
ddumitr@cs.ubbcluj.ro

Department of Computer Science
Babeş-Bolyai University, Koglniceanu no. 1
Cluj-Napoca, 400084, Romania

ABSTRACT

Probabilistic model building methods can render difficult problems feasible by identifying and exploiting dependencies. They build a probabilistic model from the statistical properties of multiple samples (population) scattered in the search space and generate offspring according to this model. The memory requirements of these methods grow along with the problem size as the population must be large enough to guarantee proper initial-supply, decision-making and accurate model-building.

The paper presents a novel model based trajectory method, which samples only one point at the time and infers the problem structure *online* by means of Artificial Neural Network based machine learning technique.

As case study we show how the proposed method can very efficiently address hard, non-separable building-block problems, specially designed to be solvable only by population based recombinative methods.

The small memory requirement and fast convergence of the proposed method comes at the cost of a tradeoff: the complexity of an accurate model building is bounded by the exponential of the order of dependencies detected by the online learning.

Categories and Subject Descriptors

G.1.6 [**Mathematics of Computing**]: Global Optimization—*Analyze*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search

General Terms

Algorithms, Design, Theory

Keywords

Model based local-search, online model building, adaptive neighborhood structure

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-131-6/08/07.

1. INTRODUCTION

Estimation of Distribution Algorithms (EDAs) extend the classical framework of Evolutionary Algorithms (EAs) by building and using probability functions and generating samples accordingly, which enables them to represent and *learn* the structure between the search variables. These methods can detect dependencies spread over the entire length of the genome and solve modular problems, which are intractable using fixed, problem independent operators [12].

EDAs able to exploit hierarchical structure can solve problems with higher order dependencies, where many or even all variables may be non-linearly interdependent. However, in these problems, like the Hierarchical IFF [17] or the Hierarchical Trap function [13], the dependencies are limited such that efficient problem solving is possible due to the hierarchical module organization.

EDAs usually search for a model which fit an existing population according to some criteria, like the Minimum Description Length Principle [3] or Bayesian-Dirichlet metric [12]. The size of the population must be large enough to guarantee proper initial-supply, decision-making and accurate model-building. Model building may require many model evaluations with regard to the population, resulting in a significant computational burden which can even exceed the bound for the number of evaluations [2].

In this paper a novel Online Model Based Local-Search (OMBLS) framework is presented, similar in approach with [5] in the sense that it employs an adaptive neighborhood structure which facilitates the operation directly on modules. Nevertheless, this approach does not use a memory to store semi-converged solutions for later analysis, one point is sampled at the time and the search experience is accumulated and information about the problem structure is inferred from a single data structure.

The online learning is achieved by means of an Artificial Neural Network (ANN) model that can be efficiently trained adaptively, namely Self Organizing Maps (SOM) based on *unsupervised* learning. ANNs in general and SOMs in particular are known to have the capability to automatically learn the hidden structure of an input space; they have the ability to preprocess input patterns to produce simpler patterns with fewer components [7].

A great advantage of online learning, besides the low memory requirements, is that the model inferring process is automatic; there is no model search and repeated costly evaluation against a set of samples (population).

The following section presents hierarchically decomposable, non-separable building-block test problems. Section

3 describes the model based trajectory method with SOM based online unsupervised learning. The performance of the proposed algorithm is discussed and empirical results are presented in Section 4. Finally, the paper is concluded in Section 5.

2. THE CLASS OF HIERARCHICAL PROBLEMS

Although having a gross-scale building-block structure, hierarchical problems are hard to solve without proper problem decomposition as the blocks from these functions are not separable. These problems are formed by a hierarchy of modules where a module consists of a number of smaller, non-overlapping modules, and smallest modules at the bottom of the hierarchy are the variables of the problem. A module may be defined as a subset of the variables in the problem for which it holds that only part of the variable settings are near-optimal for some context setting [2].

The fundamental of hierarchically decomposable problems is that there is always more than one way to solve a (sub-) problem [16] leading to the separation of building-blocks "fitness" i.e. contribution to the objective function, from their meaning. This conceptual separation induces the non-linear dependencies between building-blocks: providing the same objective function contribution, a building-block might be completely suited for one context whilst completely wrong for another one. Thus the "fitness" of a building-block can be misleading if it is incompatible with its context. However, the contribution of the building-blocks indicate how can the dimensionality of the problem be reduced by expressing one block in a lower level as one variable in the upper level. Recursively forming higher order modules from lower level ones reduces the problem dimensionality; if the problem is decomposable the number of optimal settings for a module is lower than its total number of settings [15].

Hierarchical problems are very hard for mutation based hill-climbers as they exhibit a fractal like structure in the Hamming space with many local optima [18]. This bit-wise landscape is fully deceptive; the better is a local optimum the further away is from the global ones. At the same time the problem can be solved quite easily in the building-block or "crossover space", where the block-wise landscape is fully non deceptive [16]. The forming of higher order building-blocks from lower level ones reduces the problem dimensionality.

In this paper three standard and well known hierarchical test functions are used: the hierarchical IFF [16], the hierarchical XOR [17] and the hierarchical trap function [13]. These problems are defined on binary strings of the form $x \in \{0,1\}^{k^p}$, where k is the number of sub-blocks in a block, and p is the number of hierarchical levels. The meaning of sub-blocks is separated from their fitness by the means of a boolean function h, which determines if the sub-block is valid in the current context or not. In the shuffled version of these problems the tight linkage is disrupted by randomly reordering the bits. The functions with their particularities are detailed as follows.

2.1 Hierarchical if and only if (hIFF)

The hIFF has $k=2$ and it is provided by the if and only if relation, or equality. Let $L = x_1, x_2, \ldots, x_{2^{p-1}}$ be the first half of the binary string x and $R = x_{2^{p-1}+1}, x_{2^{p-1}+2}, \ldots, x_{2^p}$ the second one. Then h is defined as:

$$h_{iff}(x) = \begin{cases} 1 & \text{, if } p = 0; \\ 1 & \text{, if } h_{iff}(L) = h_{iff}(R) = 1 \text{ and } L = R; \\ 0 & \text{, otherwise.} \end{cases} \tag{1}$$

Based on h_{iff} the hierarchical iff is defined recursively:

$$H_{iff}(x) = H_{iff}(L) + H_{iff}(R) + \begin{cases} length(x), & \text{if } h_{iff}(x) = 1; \\ 0 & \text{, otherwise.} \end{cases} \tag{2}$$

At each level $p > 0$ the $H_{iff}(x)$ function rewards a block x if and only if the interpretation of the two composing sub-blocks are both either 0 or 1. Otherwise the contribution is zero.

The hIFF has two global optima: strings formed only by 0's or only by 1's. At the lowest level the problem has $2^{l/2}$ local optima where l is the problem size.

2.2 Hierarchical XOR (hXOR)

The global optima of hIFF are formed by all 1's or all 0's, which may ease the task of some methods biased to replicate particular allele values. To prevent the exploitation of this particular problem property the hXOR was designed [17]. This problem is much more difficult due to its reduced potential for exploiting repetitiveness.

The definition of hXOR is analogous with the hIFF, having only a modification in the validation function h, where instead of equality we do a complement check:

$$h_{xor}(x) = \begin{cases} 1 & \text{, if } p = 0; \\ 1 & \text{, if } h_{xor}(L) = h_{xor}(R) = 1 \text{ and } L = \bar{R}; \\ 0 & \text{, otherwise.} \end{cases} \tag{3}$$

$\bar{R}$ stands for the bitwise negation of R.

The two global optima of hXOR are composed by half zeros and half ones. Having the same problem structure, one would expect that an algorithm which applies problem decomposition to perform equally well on both problems. As already mentioned, this is not always the case as some methods may be biased to replicate particular alleles, solving the hIFF in an easier manner.

2.3 The Hierarchical Trap Function (hTrap)

The underlying structure of the hTrap is a balanced k-ary tree, where $k \geq 3$. Blocks from lower level are interpreted by a *mapping function* similar to the one from the hIFF: a block of all 0's and 1's is mapped to 0 and 1 respectively, and everything else is interpreted as '-' or *null*.

The contribution function is a trap function of unitation (its value depends only by the numbers of 1's in the input string) of order k, based on two parameters f_{high} and f_{low} which define the degree of deception.

Let u be the unitary of the input string. Then the trap function is defined as:

$$trap_k(u) = \begin{cases} f_{high} & \text{, if } u = k; \\ f_{low} \times \frac{k-1-u}{k-1} & \text{, otherwise.} \end{cases} \tag{4}$$

If any position in the input string is null ('-') then the contribution is zero.

In this paper we use hTrap function based on $k=3$ and f_{high} and f_{low} set to 1 for all except the highest level. The

decision between competing BBs can be carried out only on the highest level, where $f_{high} = 1$ and $f_{low} = 0.9$.

In the next section we detail the Online Model Based Local-Search (OMBLS) method, which will be used to address these hierarchical problems.

3. ONLINE MODEL BASED LOCAL-SEARCH

In Model Based Local-Search [14, 11, 5] the changing of the fixed problem representation with a problem structure aware description, leads to an efficient modular search and can even lead to the solving of hierarchical problems by repeated decomposition.

In non-separable building-block problems, to reduce the search space and make the search efficient, the particular modular structure of the problem must be detected and the representation of the solution prototype must be evolved to reflect the gained knowledge about the building-blocks and their context-optimal settings. Promising sub-solutions must be kept until the method advances to upper levels where a correct decision can be made

The adaptive representation which express detected modules as new variables of the search is the key to conquer hierarchical problems: by exploring the neighborhood of the current module configuration the next level of modules can be revealed.

3.1 Module aware representation

Let us denote the current module knowledge at state s by

$$M(s) = (m_1, m_2, \ldots, m_n) \tag{5}$$

where m_i-s are the modules or building-blocks and n is the number of detected modules.

Each module m_i can have multiple configurations relating to different context-optimal settings:

$$V_i = \{v | v \in \{0,1\}^l\} \tag{6}$$

where l is the length of m_i. This allows the sustenance and parallel processing of competing context-optimal schemata.

The current state s is formed by particular context-optimal settings of the known modules:

$$s = (v(m_1), v(m_2), \ldots, v(m_n)) \tag{7}$$

where n is the number of known modules and $v(m_i) \in [1, |V_i|] \cap \mathbb{N}^*$ gives the index of a candidate configuration of the building-block m_i from the set V_i. For example, having $n = 3$ the state $s = (1, 2, 1)$ is translated as being formed from the combination of the first context-optimal setting of module one, the second context-optimal setting of the second module and the first candidate configuration of the third module.

In the case of binary problems, the representation is initialized with each variable as a basic module m_i. The initial V_i context-optimal settings for each basic module are $\{0, 1\}$.

3.2 Employed local-search

The modules of the hierarchical problems under investigations have only two context-optimal settings at each level. As the module sizes are small (pairwise combinations in the case of hIFF and hXOR, respectively combination of three elements in the case of hTrap) one of the two context-optimal setting (locally optimal) is easily reached from a randomly initialized module configuration, requiring at maximum the setting of one variable to the value that will form a context-optimal setting.

As locally optimal setting of modules are in immediate vicinity of randomly initialized module configurations, we use a simple greedy search among these configurations, alike the method used in [5]. Nevertheless, one should note that for harder problems, where context-optimal setting are harder to reach a more powerful local-search method is required. A model based local-search, built upon macro-mutation [9] search, had shown the ability to identify and exploit much larger module sizes [6].

The building-block hill-climbing employed in this paper is rather straightforward: instead of flipping bits, the search focuses on the best local context-optimal building-block configuration. Each module is processed systematically by testing its configurations and selecting the one which provides the best objective function value. While the search for the optimal configuration of a particular block is carried out, the configurations of the other building-blocks are hold still. The search for context-optimal module settings can be schematized as follows:

Function `MWGS(M,V)`

```
   /* Generate a random state s according to the
      current building-block knowledge M(s)   */
1  s ← RandomState(M);
   /* Randomly permute the order of modules and
      their related settings for unbiased greedy
      search.                                 */
2  [MR VR] ← RandPerm(M,V);
3  foreach module mi from MR do
4      foreach setting vj from VR_mi do
5          set v(mi) in s to vj;
6          if the change results in a decrease of the
           objective function then
7              undo change;
```

3.3 Learning the structure within OMBLS

When applying the greedy search on the presented hierarchical problems, the method will always discover a context-optimal setting for each module, but due to the non-linear interdependencies between the modules organized hierarchically, it will converge to a local optima in most of the cases.

As there are only two context-optimal settings for each module, the variables of the converged solutions will be grouped in a subspace which has lower dimensionality than the dimensionality of the data. Certain ANN models have the ability to adaptively process input patterns and to produce simpler patterns with fewer components in accordance with the topology of the input space. On longer term, by training the network with multiple solutions, the modular structure of the problem can be inferred.

A network which seeks to preserve the topological properties of the input space is the Self Organizing Map (SOM) [10]. The network is trained using *unsupervised learning* to produce a two dimensional, discretized representation of the input space, called a map.

The SOM weight adapting algorithm is based on the competitive learning paradigm; vector quantization is used to model probability density functions by the distribution of prototype vectors. Concretely, when a sample s is presented to the network, the following steps are executed:

1. The Euclidean distance to all weight vectors is computed.
2. The neuron with weight vector most similar to the input is nominated as the best matching unit (BMU).
3. The weights of the BMU and neurons in the neighborhood are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the BMU.

This algorithm can be very well iteratively updated online with "live" data, directly inputting the results (locally converged states) of the model based local-search, rather than training with samples from a memory.

In the proposed method we use a SOM with a lattice of $5 * 5$ neurons, which are arranged in a rectangular grid with regular spacing. A weight vector of size n where n is the number of known modules and a position in the map space is associated with each neuron. To induce a symmetric negative bias into the adjustment of the weights, the network is trained with half of the values from the input ranges moved into the negative domain. Thus, the two possible values of a variable are inputted as $\{-1, 1\}$, for a V_i with cardinality of three the inputs are recoded as $\{-1, 1, 2\}$ and so on.

The network can be trained with data from an apriori settled number of epochs or one can use a dynamic stopping condition which checks if the change in the values of weights in the last few epochs is bellow a certain threshold.

After training the SOM online, dependencies are deduced form the internal representation of the network based on the heuristic that similar inputs should produce similar patterns in their associated weights i.e dependent inputs have roughly the same values for their weights.

Accordingly, we use the following metric for detecting the dependency between variables x_i and x_j:

$$d(x_i, x_j) = \sum_{l=1}^{r} ||W_{il}| - |W_{jl}||^2 \quad (8)$$

where r is the number of neurons on the rectangular lattice and W represents the weights of the SOM. This relation measures the closeness of different variables by taking into account the weights related to them.

We consider x_i and x_j as being dependent if the following relation holds for a predefined ϵ threshold.:

$$d(x_i, x_j) \leq \epsilon \quad (9)$$

Dependent modules will be merged into the same composite module for the next phase of the search.

Analyzing the weights of the network we are able to decide which variables are linked but in order to collapse the search space we need their context-optimal setting also. As a consequence, provided with only the variable relationships, for each new composite block the method exhaustively evaluates all the possible combinations of sub-modules in the context of randomly generated states and retain the best λ ones.

The module knowledge is updated with $M(s)$ containing the modules according defined by relation 9, whilst the context-optimal settings V_i of each new module is filled with the λ best configurations found by the exhaustive search.

The following section summarizes the Online Model Based Local-Search.

Algorithm 2: Online Model Based Local-Search

```
Data: M, V, n_S, λ, ε, @stopping_cond.
while not @stopping_cond do
    /* PHASE I                                          */
    /* Build the SOM                                    */
    net ← InitializeSOM|M(s)|);
    for i = 1, n_S do
        /* Generate a random state s according to
           the current building-block knowledge
           M(s)                                         */
        s ← RandomState(M);
        /* Apply module-wise search                     */
        s ← MWGS(M, V);
        /* Train the network online using vector
           quantization                                 */
        net ← Train(net, s);
    /* PHASE II                                         */
    /* Detect possible modules via weight
       analysis                                         */
    nm ← GetLinkages(net.Weights, ε);
    /* Identify best settings for new modules by
       exhaustive search                                */
    CO_set ← SearchForBestSettings(nm, λ);
    /* Collapse the search space and update the
       building-block configuration according to
       the detected modules and their
       context-optimal settings                         */
    [M, V] ← UpdateModuleKnowledge(nm, CO_set);
```

3.4 OMBLS algorithm

Starting from a representation in concordance with the original problem, in a first phase search experience is accumulated by training the SOM online with the "live" states provided by repeated local-search working on the current representation. The local-search strategy used must be powerful enough to discover fully optimized modules at a single hierarchical level.

After convergence of the network, in the second phase the structure of the input space is inferred from the weights of the network and expressed by variable linkages. Furthermore, an exhaustive search is performed according to the detected linkages, to find the best context-optimal settings for each new module.

The search space is collapsed as the module aware representation is updated according to the detected linkages and their most fit context-optimal settings.

After these steps, the search enters again phase one, with the local-search operating on the newly derived representation, further exploring their combinative neighborhood.

The method stops when the search space can not be collapsed anymore or a predefined number of objective function evaluations is exceeded.

Formally the OMBLS is outlined in Algorithm 2.

4. PERFORMANCE

Assuming that the used machine learning technique successfully detects the correct dependencies, the global convergence of the OMBLS on the studied test suites can be proven by showing that there is a path towards the global optima, easily followed by the method.

As mentioned in Section 3.2, on the test suites under investigation the numbers of sub-blocks in a block at each hierarchical level is maximally three and there are two context optimal settings for each module. If the partial knowledge about the problem structure is correct, starting from a random combinations of sub-blocks, if the global optima have not yet been attained, one of the two context-optimal setting at the next level is always reached by the greedy search. As fully-optimized modules are found they are expressed as variables in the next phase of the search, which leads to a simple recursive solving of all hierarchical levels as in [5].

In the case of hIFF and hXOR proving that the greedy search will find the variable and its settings that are context-optimal is trivial. Let m_1 and m_2 be two dependent modules, each with two context-optimal settings $V_i = \{1, 2\}$. No matter which random combination of module setting we take from $C_{m_1,m_2} = \{(1,1),(1,2),(2,1),(2,2)\}$ possible ones and which module relation rule we apply (IFF or XOR), it is obvious that by testing all possible values for m_1 one will match up with the setting of m_2 and will increase the number of fully-optimized modules.

The same reasoning extends to hTrap also. Having three modules each with two context-optimal setting and supposing without loosing generality that the two context-optimal setting at the next level are given by $c_1 = \{1, 1, 1\}$ and $c_2 = \{2, 2, 2\}$, when randomly choosing a combination of modules we can have the following situation:

- the random combination is c_1 or c_2 – we are done;
- we have a combination containing two "1's" and one module with setting "2"; when the greedy search sets the module setting from "2" to "1" a fitness increase will signal the formation of a new fully optimized module and the new state is accepted;
- we have a combination containing two "2's" and one module with setting "1"; same reasoning as above apply.

Provided that the SOM indicates the correct dependencies, the correct partial knowledge about the problem structure is guaranteed by the fact that the methods performs an exhaustive search in order to determine the most suitable settings for each module.

Due to the deterministic nature of the greedy search and following the above mentioned path, a tight upper bound on the performance of OMBLS can be given.

At one hierarchical level, the module-wise greedy search will use a number of objective function evaluations in concordance with the number of modules l, their context-optimal setting λ and number of epochs used to feed the network n_S:

$$T_{MWGS} = n_S \cdot \lambda \cdot l \tag{10}$$

The search for the context-optimal settings will take a number of objective function evaluations exponential in the size (denoted by k_i) of the newly discovered modules M' and the number of context-optimal settings:

$$T_{COS} = \sum_{i=1}^{|M'|} \lambda^{k_i} \tag{11}$$

If we have p hierarchical levels and the number of sub-modules in each composite module is k, the upper bound of

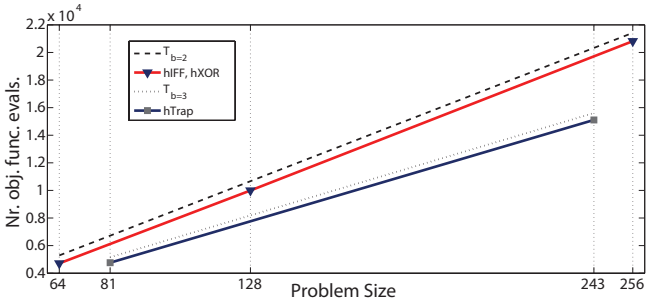

Figure 1: Scaling of OMBLS with module-wise greedy local-search strategy on hIFF, hXOR and hTrap.

the whole method is given by the summation of the greedy search T_{MWGS} and the search for context-optimal settings T_{COS} on each hierarchical level:

$$T = \sum_{\substack{l=k \\ l=l*b}}^{k^p} (n_S \cdot \lambda \cdot l + \frac{l}{k} \cdot \lambda^k) \tag{12}$$

To empirically confirm this result and to test the efficiency of the SOM based online linkage detection technique, the scalability of the OMBLS have been tested on 64-bit, 128-bit and 256-bit shuffled hIFF and hXOR problems, respectively on 81-bit and 243-bit shuffled hTrap problem instances. A total number of 100 independent runs were averaged.

As the samples inputted to the network are high-quality converged states, we restricted the number of module-wise local-search epochs to $n_S = 20$. The settings for other parameters were $\lambda = 2$ and $\epsilon = 1.0e-8$.

The method found one of the global optima in all cases confirming the efficiency of the online SOM based linkage learning.

The scaling of the method on the different test suites is depicted in Fig. 1 along with the upper bound given by the base module sizes. As results on hIFF and hXOR were very similar due to the identical problem structure (complete binary tree), they are depicted in a single graph.

Potentially, the most costly operation of OMBLS is represented by the search for context-optimal settings, which is exponential in the order of dependencies revealed by the SOM based learning. This phenomena is not particular for our method. The size of population in EDAs (implicitly the number of objective function evaluations in each generation) is also lower bounded by the exponential of the order of dependencies covered by the probabilistic model.

Nevertheless, for boundedly difficult problems, when the order of dependencies is low compared to the problem size ($k << n$) as in the problems studied here, the computational cost of the search for context-optimal settings can be approximated with a constant. Then, the total cost is dominated by the computational burden of the employed local-search. In our case, as the greedy search is linear in the number of modules, from Eq. 12 results a very efficient sub-linearithmic running time, confirmed empirically by our experiments (see Fig. 1).

The memory requirements of the OMBLS are very low, being linear in the problem size.

5. CONCLUSIONS AND FURTHER WORK

The paper presents a model based trajectory framework, namely the Online Model Based Local-Search (OMBLS) that learns the problem structure online by means of topology preserving SOMs. OMBLS operates via hierarchical decomposition, detected modules are used to collapse the search space and reformulate the optimization problem with discovered modules and their context-optimal settings as new search variables.

The continuous update of the module knowledge and representation of the solution prototype implicitly results in the adaptation of the neighborhood structure to the combinative neighborhood of the current building-blocks. This is why the OMBLS can very efficiently address hard, non-separable building-block problems, specially designed to be solvable only by population based recombinative methods. In these problems, modeled after the intuition of the Building Block Hypothesis [4], moving the search to the combinative vicinity of the current module aware representation facilitates the discovery of new blocks, as the Building Block Hypothesis implies that low-order building-blocks can be combined to form higher-order ones.

As training is done online, the memory requirements of the method are limited to storing one solution at the time, the knowledge about modules and a SOM which is also proportional with the number of input variables. Nevertheless, the network can only reveal variable dependencies; a further search for context-optimal settings must be employed, resulting in a model building complexity bounded by the exponential of the order of dependencies detected by the online learning. This computational cost is similar with the population requirement of classic EDAs, where the number of samples is also lower bounded by the exponential of the order of dependencies covered by the probabilistic model.

If context-optimal settings can be discovered by a module-wise local-search strategy in linear time and the order of dependencies is limited to small k, the proposed framework holds a qualitative advantage over other methods as it scales at most linearithmicaly.

We look to exploit the fast automatic model building and low memory requirement of the proposed methods on problems with million(s) of variables, where the classical solving with population based methods would imply very large memory requirements and huge computational costs for model building.

Acknowledgments

This work was sponsored by MEdC-CNCSIS and by the Sapientia Institute for Research Programs (KPI).

6. REFERENCES

[1] D. Dasgupta and Z. Michalewicz, editors. *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[2] E. D. de Jong, R. A. Watson, and D. Thierens. On the complexity of hierarchical problem solving. In H.-G. Beyer and U.-M. O'Reilly, editors, *GECCO '05*, pages 1201–1208. ACM, june 2005.

[3] P. Grünwald, I. J. Myung, and M. Pitt. *Advances in Minimum Description Length*. MIT Press, Apr. 01 2005.

[4] J. H. Holland. *Adaptation in natural artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[5] D. Iclanzan and D. Dumitrescu. Overcoming hierarchical difficulty by hill-climbing the building block structure. In D. T. et al., editor, *GECCO '07: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation*, volume 2, pages 1256–1263, London, 7-11 July 2007. ACM Press.

[6] D. Iclanzan and D. Dumitrescu. Going for the big fishes: Discovering and combining large neutral and massively multimodal building-blocks with model based macro-mutation. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and Evolutionary Computation*. ACM Press, 12-16 July 2008. (accepted).

[7] J. Jiang. Image compression with neural networks: A survey. *SP:IC*, 14(9):737–760, July 1999.

[8] R. Johnston and H. Cartwright, editors. *Applications of Evolutionary Computation in Chemistry*. Springer, 2004.

[9] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, NM, 1995.

[10] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[11] C. F. Lima, K. Sastry, D. E. Goldberg, and F. G. Lobo. Combining competent crossover and mutation operators: a probabilistic model building approach. In *GECCO '05*, pages 735–742, NY, USA, 2005. ACM.

[12] M. Pelikan. *Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms*. Springer Verlag, 2005.

[13] M. Pelikan and D. E. Goldberg. Escaping hierarchical traps with competent genetic algorithms. In L. S. et al., editor, *GECCO '01:*, pages 511–518, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

[14] K. Sastry and D. E. Goldberg. Designing competent mutation operators via probabilistic model building of neighborhoods. In *GECCO '04*, pages 114–125. Springer, LNCS, vol. 3103, June 26–30, 2004.

[15] R. A. Watson. *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, Apr. 01 2002.

[16] R. A. Watson, G. Hornby, and J. B. Pollack. Modeling building-block interdependency. In *PPSN V: Proc. of the 5th International Conference on Parallel Problem Solving from Nature*, pages 97–108, London, UK, 1998. Springer, LNCS.

[17] R. A. Watson and J. B. Pollack. Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In S. Brave, editor, *GECCO '99: Late Breaking Papers*, pages 292–297, Orlando, Florida, USA, 13 July 1999.

[18] R. A. Watson and J. B. Pollack. Symbiotic composition and evolvability. In J. Kelemen and P. Sosik, editors, *Advances in Artificial Life, 6th European Conf., (ECAL 2001)*, pages 480–490, Berlin, 2001. Springer.