# Going for the Big Fishes: Discovering and Combining Large Neutral and Massively Multimodal Building-Blocks with Model Based Macro-Mutation

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

A major challenge in the field of metaheuristics is to find ways to increase the size of problems that can be addressed reliably. Scalability of probabilistic model building methods, capable to rendering difficult, large problems feasible by identifying dependencies, have been previously explored but investigations had mainly concerned problems where efficient solving is possible with the exploitation of low order dependencies. This is due to the initial-supply population sizing, where the number of samples is lower bounded by the exponential of the order of dependencies covered by the probabilistic model. With an exponentially growing population, the impact of the model building on the overall complexity, can easily exceed the bound for the number of evaluations.

In this paper we present a competent methodology, capable of efficiently detecting and combining *large* modules, even in the case of unfavorable genetic linkage and no intra-block fitness gradient to guide the search or deceptiveness. This is achieved by investing the function evaluations in a model based local-search with strong exploratory power and restricting the model building to a relatively small number of semi-converged samples.

## Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search

## General Terms

Algorithms, Design, Theory

## Keywords

Model based local-search, adaptive neighborhood structure, scalability, macro-mutation

## 1. INTRODUCTION

A critical issue with complex real-world problems is the scalability problem [19]. Accordingly, a major challenge in the field of metaheuristics is to find ways to increase the size of problems that can be addressed reliably.

The feasible sizes of problems strongly depend on their particular structures that can be used to bias the search towards high quality solutions. Without such properties to be exploited, the search space requires a systematic exploration. A problem property which can render large problems feasible when identified is that of dependencies.

Estimation of Distribution Algorithms (EDAs), an extension of Evolutionary Algorithms (EAs), use probability functions instead of a population of configurations, which enables them to represent and *learn* the structure between the search variables. If enough sampling is provided, EDAs can detect dependencies spread over the entire length of the genome and solve modular problems which are intractable using fixed, problem independent operators and representations.

Albeit, scalability of EDAs have been previously investigated [16, 1], investigations had mostly concerned boundedly-difficult problems where efficient solving is possible with the exploitation of low order dependencies. Typically, the addressed order of dependencies, which we denote by $k$, was usually less or equal then 6.

This is due to the initial-supply population sizing [2], where the number of samples in an EDA is lower bounded by the exponential of the order of dependencies covered by the probabilistic model, being $\Omega(2^k)$ in the case of binary encoding. With an exponential growth of the population, the impact of the model building on the overall complexity can easily exceed the bound for the number of evaluations.

The objective of this paper is to present a competent methodology which can detect and combine larger modules with sizes up to 8-16, in the case of unfavorable genetic linkage and no intra-block fitness gradient to guide the search (neutrality) or even deceptiveness. This is achieved by investing the function evaluations in a model based local-search with strong exploratory power, instead of using huge populations in order to supply the initial building-blocks. The dependencies are detected from only a few (semi-) converged states, using a novel Artificial Neural Network based machine learning technique.

The method can discover $m$ modules of order $k$ characterized by neutrality efficiently, by repeatedly investing

$c_2 \cdot n^2$ function evaluations into the model based local-search, where $n$ is the size of the problem and $c_2 \cdot n^2$ provides an acceptable approximation to $m2^k$, with $c_2 < k$.

The following section analyzes with what costs can large building-blocks be discovered at terms of model building and population sizes or function evaluations in the case of local-search. Suitable test suites are introduced in Section 3. In Section 4 the proposed method is detailed. Section 5 presents empirical results of the proposed method. Finally, the paper is concluded in Section 6.

## 2. HOW MUCH IS THE FISH?

Modular, boundedly-difficult problems must have $k$ small, relative to the problem size $n$. In order to ease up the analysis of scalability as a function of module size, in this paper we use the maximal value for $k$ of $\sqrt{n}$. Accordingly, the number of modules equals the order of the modules i.e. $n = km = k^2$, where $m$ is the total number of building-blocks on the lowest level.

### 2.1 Population Sizes and Model Building Cost in PMBGAs

In Probabilistic Model Building Genetic Algorithms (PMBGAs), the first step towards reliable and accurate problem solving is the tackling of the initial building-block supply. One must provide the minimum population size required to ensure the presence of at least one copy of all raw schemata.

In [4] the authors have developed facetwise models for ensuring building-block supply in the initial population. For binary alphabets, the predicted population size required to ensure the presence of all competing building-blocks with a tolerance of $\epsilon = 1/m$ is given by

$$N = 2^k(k + log(m)) \tag{1}$$

where k is the order of the modules, and m is the total number of building-blocks.

Please note that this is the minimal population size, assuring the presence of one copy of each building-block with a high probability. Actual population sizes needed to filter out noise and guarantee a proper decision making between the correct building-blocks and their competing schemata, are much larger; a correct population size is further influenced by a problem specific constant [5].

EDAs usually employ a search for a model that best fits the population according to the maximum likelihood. If we only check pairwise relation between variables and confront them against the N samples in the population we have a minimal complexity for model building of

$$T = \Omega(n^2 N) \tag{2}$$

Again, actual model building methods may be way more costly, being for example $O(l2^l n^2 N + ln^3)$ in the case of the well known BOA [15], where $l$ is the limit on the incoming edges in the case of each node.

From Eq. 1 and Eq. 2 and by setting $n = k^2$, results a lower bound on model building complexity in PMBGAs according to $k$ of

$$T = \Omega(k^4 2^k(k + log(k))) \tag{3}$$

As $k$ gets larger, the computational burden of the model building implies millions and billions of operations at every single generation, having an impact on the overall complexity that significantly exceeds the bound for the number of objective function evaluations. Even a total number of objective function evaluations of the orders of millions is vanishingly smaller than the complexity of the model building in a single generation.

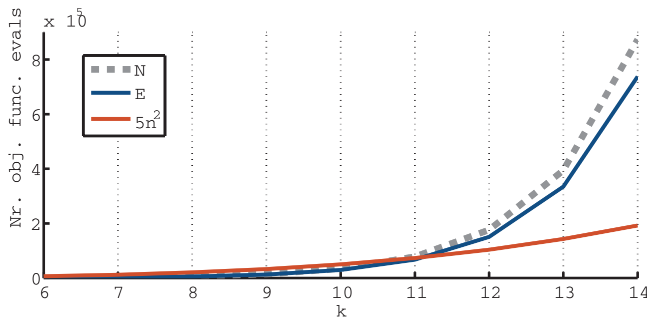

**Figure 1: The number of objective functions evaluations for the minimal population $N$, for the expected time of discovery of modules by RMHC $E$. The function $5n^2$ provides a relatively tight upper bound for these functions up to $k = 11$.**

### 2.2 Module Discovery by local-search

Another approach that can be applied and avoids the use of populations, is the discovery of building-blocks with a more systematic exploration of the search space by local-search methods. Following the fitness gradient, local-search method are able to quickly identify local optima.

What about the difficult cases when there is no intra-block fitness gradient to guide the search? What is the expected time to discover the modules by performing a random walk on a landscape dominated by neutrality?

Mitchel et. al [13] presented results for the expected time $E(k, m)$ of a Random Mutation Hill-Climber (RMHC), which mutates one randomly chosen locus at the time and accepts states with equal or higher fitness, to discover $m$ neutral blocks of order $k$. A Markov-chain analysis yielded an expected time to discover the first block $E(k, 1)$, slightly larger then $2^k$, converging towards $2^k$ as $k \to \infty$ [13]. The first block can be discovered in $E(k, 1)$, but the next ones will take longer and longer times, because as new modules are discovered, the probability of mutation to destroy already formed blocks increases. Taking this into account, the expected value for $E(k, m)$ derived in [13] is $E(k, 1)m(log(m) + \gamma)$, where $\gamma$ is Euler's constant.

Returning to the assumption that $k = m$ and approximating $E(k, 1)$ by $2^k$ we get the following expected time for the RMHC to discover the modules

$$E = 2^k k(log(k) + \gamma) \tag{4}$$

In practice, $E$ can be significantly reduced around the value $m2^k$ by using a smarter local-search that analyzes major fitness variance, following the idea introduced in [9]. In problems where modules are sharply defined, any decrease in objective function value is due to the destruction of some building-blocks. Systematically analyzing the fitness variance, one can build a preliminary model about the already formed modules and bias the mutation in such way to protect the destruction of these blocks.

What is interesting to note is that for $k \geq 4$ $E$ is smaller then $N$, the lower bound of the population size from Eq. 1. Thus, performing a local-search instead of using and evaluating large populations is also beneficial in terms of objective function evaluations. Figure 1 presents the scaling of $E$ and $N$ in function of $k$.

For moderate sizes of $k$, which nevertheless are large enough to provide real interest, the exponential growth of $2^k$ can be still tackled with reasonable computing effort. In Figure 1 the scaling of the polynomial $5n^2$ compared to $E$ and $N$ is depicted. One can notice that for $k \leq 11$, $5n^2$ provides a relatively tight upper bound and thus a reasonable approximation to $E$.

This observation facilitates the solving of problems with larger modules efficiently in the following way:

1. Local-search is repeatedly employed for a bounded number of times, to find solutions of high fitnesses containing converged modules.

2. The samples from step (1), which are in small number but already contain *correctly converged modules*, are used to *quickly* build a model which express dependencies.

3. The search is continued from step (1) with the local-search operating according to the learnt model, thus taking dependencies into account and always expressing learnt modules. In this way module combinations are explored.

The model building from only a few samples is feasible only if step (1) is successful in discovering the correct building-blocks. While PMBGAs need huge populations in order to provide the initial large modules by random sampling only (no fitness function guidance employed – which gives high tolerance to noise) and to decide between the correct modules and their most competing schemata, in the proposed method all this is handled by the local-search. The number of samples must be just large enough to make possible the demarcation of different modules. Hence, the *quick* and correct model building from a restricted (but high quality) samples is an achievable approach. Nevertheless, while much quicker this approach is less tolerant to noise.

The RMHC is not able to escape local optima as it has a restricted neighborhood structure, by mutating only one allele at the time. In our proposed method we use a local-search based on an operator that mutates several positions at the time, namely the Macro-Mutation [11]. This operator enables large jumps in the search space, thus it has a great exploratory power and the potential to escape local optima.

The exact expected runtime for the macro-mutation algorithm to discover neutral blocks is unknown, we use the results from the RMHC analysis as an estimate. As we will later see in Section 4.3.2 as noise filtering is used, in order to succeed, it is enough if the local-search discovers the blocks in most of the cases; complete convergence of all blocks every time is not mandatory.

## 3. TEST FUNCTIONS

### 3.1 Extended Shuffled Royal Road Function

The first function may be regarded as an extension of the "Royal Road" (RR) function introduced in [12]. RR functions consist of a set of schemas (building-blocks) where each block confers a fitness contribution at one isolated optimal peak on an otherwise flat landscape; particular pairwise hierarchical combinations of blocks are further rewarded. Nevertheless, RR does not exhibit local optima because the blocks are separable, each block at the base level having only one optimal setting [21].

In order to introduce module interdependence, we modify the RR similarly with the hierarchical problems [20], by introducing two context optimal setting for each module. The main difference between our function and the ones presented in [21] or [14], is that the proposed function is not *hierarchically consistent* [21]. Albeit, some blocks are certainly hierarchically grouped, it is not mandatory to combine all blocks at all hierarchical levels. The fully hierarchical pairwise combination and reward of modules, would require the number of base modules to be a power of 2. As we want the number of modules to equal the order of the blocks this is not achievable.

A more formal definition of the Extended Shuffled Royal Road (ESRR) function follows. The function involves an even number of $k$ blocks of order $k$, $k \geq 8$, randomly scattered along the input space. The set of indexes defined by blocks $b_i$ $i = \overline{1,k}$ are disjunct, not overlapping ones: $b_i \bigcap b_j = \emptyset$ if $i \neq j$. The fitness of the blocks are neutral, providing a fitness improvement only at two isolated optimal peaks on an otherwise flat landscape.

Let us denote by $u(s)$ the unitary (the numbers of ones) of a binary string $s$. Then, the fitness of a block $b$ in a binary string $x$ is given by:

$$\sigma(x,b) = \begin{cases} 1 & \text{, if } u(x[b]) = 0 \text{ or } u(x[b]) = length(b); \\ 0 & \text{, otherwise.} \end{cases} \tag{5}$$

where $\sigma$ rewards uniform blocks of all 0's or all 1's.

The fitness of the entire function is given by the summation of the fitnesses provided by each module and of the particular hierarchical combinations of blocks.

$$\begin{aligned} ESRR_k(x) \quad = \quad & r_1 \sum_{\substack{1 \leq i \leq k \\ i = i+1}} \sigma(x, b_i) + r_2 \sum_{\substack{1 \leq i < k \\ i = i+2}} \sigma(x, b_i \bigcup b_{i+1}) + \\ & r_3 \sum_{\substack{1 \leq i < k-2 \\ i = i+4}} \sigma(x, b_i \bigcup b_{i+1} \bigcup b_{i+2} \bigcup b_{i+3}) \end{aligned} \tag{6}$$

where $r_1$ is the bonus for a correctly detected base module, $r_2$ rewards successful pairwise combination of blocks while $r_3$ recompenses the formation of a valid quadruplet composite module.

Note that for a number of base modules not divisible by four, $k \neq 4l$, there are blocks that are not rewarded at the third hierarchical level, indifferent of their setting. For example, for $k = 10$ the last two blocks are not taken into account at this level, as there are no other blocks which whom they could form a quadruplet composite module.

### 3.2 Hierarchical Massively Multimodal Deceptive Function

To test how the proposed method can handle deceptiveness and multimodality, we regard the extension of the bipolar deceptive function proposed by Goldberg et al. [3].

The function proposed in [3] is defined on binary strings of length 6 and is both deceptive and multimodal as it has two global optima at strings formed by all 0's or all 1's and a local

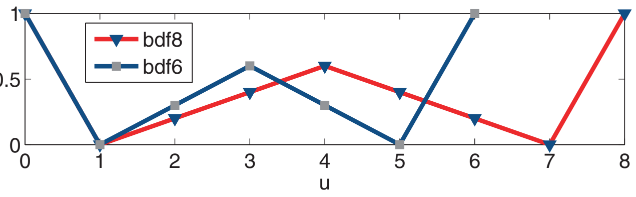

**Figure 2: Construction of bipolar deceptive functions of order $k$ by placing a local optima to $u = \frac{k}{2}$.**

optima placed where the unitary of the block equals 3. There are many points belonging to the deceptive attractor as there are $\binom{6}{3}$ ways to get half zeros half ones in a binary string of length 6 and only two ways to render a global optima. Concatenating just 6 of these functions will render more then a hundred million of locally optimal points.

We consider the extension of the simply concatenated version of bipolar deceptive functions in the following way: a hierarchical consistent version of the problem is used, where the modules combination (if all found) are also rewarded according to a bipolar deceptive function. Furthermore, the problem is shuffled by randomly reordering the bits, mixing the bipolar deceptive blocks on all levels. Finding the right module combination is very difficult, the fitness signal on the second level being again average case misleading and strongly multimodal.

The proposed Hierarchical Massively Multimodal Deceptive (HMMD) function involves an even number of $k$ disjunct, randomly mixed blocks of order $k$. The fitness of these base modules is given by a bipolar deceptive function. Let again, $u(s)$ denote the unitary of a binary string $s$. Then, the bipolar deceptive function of order $k$, for the bits defined by a block $b$ in a binary string $x$, is given by:

$$bdf_k(x,b) = \begin{cases} 1, \text{ if } u(x[b]) = 0 \text{ or } u(x[b]) = k; \\ 0.6 \cdot \left(1 - \frac{||\frac{k}{2} - u(x[b]||}{\frac{k}{2} - 1}\right), \text{ otherwise.} \end{cases} \quad (7)$$

In this paper we will extend our analysis to HMMD functions with base modules of size eight also, as we are mainly interested in the possibilities for the discovery of larger blocks. Figure 2 depicts the bipolar deceptive functions of order 6 and 8.

HMMD is a hierarchically consistent problem, meaning that "the nature of the problem is the same at all levels in the hierarchy" [21]. In order to be able to evaluate modules according to the same bipolar deceptive function criteria, we need a transform function defining the "meaning" of each block. The transform function creates from a module of order $k$ a symbol following the rule:

$$t(x,b) = \begin{cases} 1 & \text{, if } u(x[b]) = k; \\ 0 & \text{, if } u(x[b]) = 0; \\ null & \text{, otherwise.} \end{cases} \quad (8)$$

where *null* denotes non-solutions which are undesirable at the next hierarchical level.

Using the above defined transform function to decode the base modules into a string $y$ of length $k$, the evaluation function rewarding combinations of modules at the second hierarchical level is given by:

$$wr_k(y) = \begin{cases} 0 & \text{, if } y \text{ contains } null; \\ bdf_k(y, \{1 \dots k\}) & \text{, otherwise.} \end{cases} \quad (9)$$

This function is a wrapper around the bipolar deceptive evaluation, which does not award combinations of modules if not every block is correctly identified, otherwise returning the value given by $bdf_k$.

Finally, the HMMD function with blocks of order $k$ can be defined as the summation of the fitnesses provided by each module at the base level and of their combination at the second level:

$$HMMD_k(x) = r_1 \sum_i^k bdf_k(x, b_i) + r_2 \cdot wr_k(\{t(b_1) \dots t(b_k)\}) \quad (10)$$

where $r_1$, $r_2$ controls the amount of reward at each level.

This function is highly multimodal. For example $HMMD_8$, where there are eight modules of length eight, each giving a total of $\binom{8}{4} + 2 = 72$ optima for a total of $72^8$, from which only two are global ones: strings formed by all 0's or by all 1's. The number of local optima is astronomical, being somewhere half between seven hundred trillion and three quarters of one quadrillion! Finding the one of the two global optima, from all these points where the search may potentially get stuck, can be achieved only by a proper exploration (sampling) of the search space combined with an exact problem decomposition.

## 4. MODEL BASED MACRO-MUTATION

Previous theoretical studies had shown that when identified, the local-search according to the building-block structure is more efficient than selectorecombinative GAs, on deterministic additively-separable problems of bounded difficulty [18]. Based on these findings, later developments proposed competent selectomutative GAs [17] and methods that combine both competent crossover and competent module-wise mutation operators [10], that are able to solve hard problems quickly, reliably, and accurately.

The mutation operator in these approaches use the probabilistic model of linkage groups derived from *populations*. Thus, in order to be able to discover large basic blocks, these approaches also need big population sizes.

Our approach proposes a different approach. Starting from a representation in concordance with the original problem, in a first phase, search experience is accumulated by repeated local-search working on the current representation. In a second phase the search experience is used to learn linkages and adapt the representation accordingly. After this, the search enters again phase one, but with the local-search operating on the newly derived representation. Enabled with an adaptive neighborhood structure, the proposed method is able to efficiently explore the combinative neighborhood structure of the learnt modules, thus it can discover further composite modules.

### 4.1 Module aware representation

Let us denote the current module knowledge at state $s$ by

$$M(s) = (m_1, m_2, \dots, m_n) \quad (11)$$

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:

$$V_i = \{v | v \in \{0,1\}^l\} \quad (12)$$

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:

$$s = (v(m_1), v(m_2), \dots, v(m_n)) \quad (13)$$

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 concatenation 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 $3^{rd}$ module.

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

After dependencies have been detected and new composite modules have been formed, their configurations $V_i$ are obtained by taking all the different substrings from the memory, from the positions defined by the new module. For example if a new module $m_i$ is formed on the positions $[1, 2, 3, 8]$, $V_i = unique(mem(:, [1, 2, 3, 8]))$ where $mem$ is the memory containing $n_S$ samples, and ":" denotes a for-loop iterator processing all samples from 1 to $n_S$.

## 4.2 Macro-Mutation Hill-Climber

A method with great exploratory power, the Macro Mutation Hill-Climber (MMHC) had been shown to be a very powerful hill-climbing method, which can outperform GAs even on problems where each building-block corresponds to a deceptive trap function, provided that the problem has a tight linkage [7].

In the mutation operator of the MMHC a randomly chosen number of loci from the representation vector are set to random alleles. If the fitness of the newly generated state is greater or equal as the actual one, then the new state is accepted [7].

In our method, when a (major) fitness improvement is detected, bigger than a predefined threshold $\zeta$, a systematic search analyzes every setting of each module. By recording the modules which imply the same fitness drop if their setting are changed, a preliminary model about the already formed building-blocks is obtained. This model is used to bias the otherwise random and blind loci choice for mutation: positions believed to be part of the same module are mutated always together and initially rarely. Rather, the search focuses on the subset of positions from the representation vector which are not yet classified as being part of a module, macro-mutating them as described above. When all positions are classified as part of blocks, the operator always randomly chose a preliminary module and macro-mutate on the positions that define this module.

Macro-mutation operates according to the model provided by the modules $M(s)$, randomly choosing their configuration from $V$.

## 4.3 Learning the structure

When applying the local-search to problems of interest, the method will discover some elementary modules, but due to the non-linear interdependencies, it will ultimately converge to a local optima with overwhelming probability. However, the variables of the converged solutions will be grouped in a subspace which has lower dimensionality than the dimensionality of the data, due to the modularity of the problem. On longer term, multiple stored converged solution may be used to infer the modular structure of the problem.

Feature extraction is applying a mapping of a multidimensional space into a space of fewer dimensions. Artificial Neural Networks (ANNs) are very well suited for this task as they have been shown 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 [6]. Nevertheless, one should note that other classical methods, like the ones based on Bayesian networks, can also be used to detected the modular structure from the (semi-)converged states. The computational burden of the classical model building techniques would also be moderate, as exponential populations in the block size are avoided, the number of samples being very low.

### *4.3.1 Learning with Self Organizing Maps*

In this paper we use a Self Organizing Map (SOM) to detect dependent inputs. SOMs are trained using *unsupervised learning* to produce a two dimensional, discretized representation of the input space of the training samples, called a map [8].

The interesting feature of SOMs, exploited in this paper, consist in the fact that the mapping is *topology preserving* and similar inputs tend to have similar weights.

In our approach a lattice of $5 * 5$ neurons is used, that are arranged in a rectangular grid with regular spacing. A weight vector of dimension $n$ where $n = |s|$ (one input for each known module) and a position in the map space is associated with each neuron. Also, the inputs are normalized from the $[1, |V_i|] \cap \mathbb{N}^*$ interval to a $[-a, b] \cap \mathbb{Z}^*$ interval, where $b = |V_i| - 1$, $a = b$ if $n = 2k$ and $a = b - 1$ if $n = 2i + 1$ to induce a symmetric negative bias into the adjustment of the weights. With only positive inputs, all weights would be always pushed towards positive values, making the inference of linkages by weight analysis impossible.

Dependencies are deduced form the internal representation of the SOM 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 $x_j$:

$$d(x_i, x_j) = \sum_{l=1}^{r} ||W_{il}| - |W_{jl}|| \quad (14)$$

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.

If $d(x_i, x_j) \leq \epsilon$, where $\epsilon$ is a predefined threshold, then we consider $x_i$ and $x_j$ as being dependent; they will be merged into the same module for the next phase of the search.

An important issue regards the number of samples needed to correctly delimitate the modules. We can approximate the probability that in $n_S$ samples, the settings of two uncorrelated variables always match up with $0.5^{n_S}$ for the binary case,. We use $n_S = 16$ samples, so this probability is less then 1e-4.

**Algorithm 1**: Model Based Macro Mutation

```
Data: $M, V, n_S, z, c_2, @stopping\_cond, \epsilon_1$.
while not $@stopping\_cond$ do
    $n \leftarrow |M(s)|$;
    /* PHASE I */
    for $i = 1, n_S$ do
        /* Generate a random state $s$ according to
           the current building-block knowledge
           $M(s)$ */
        $s \leftarrow RandomState(M)$;
        /* Apply macro-mutation according to the
           current model for $c_2 n^2$ evals */
        $s \leftarrow MBMM(s, [M, V], c_2 n^2)$;
        $mem[i] \leftarrow s$;
    /* PHASE II */
    $T \leftarrow NormalizeDataRanges(mem)$;
    $F \leftarrow 0_{n \times n}$;
    for $l = 1, z$ do
        $net \leftarrow InitializeSOM()$;
        /* Randomly select four samples */
        $S \leftarrow ChoseRandomSamples(T, 4)$;
        $net \leftarrow Train(net, S)$;
        /* Detect possible modules via weight
           analysis */
        $nm \leftarrow GetLinkages(net.Weights, \epsilon)$;
        /* Update the frequency matrix */
        $F \leftarrow Update(F, nm)$;
    /* Get modules from the frequency matrix */
    $b \leftarrow GetBaseModules(F, \epsilon_1)$;
    /* Merge overlapping modules */
    $b \leftarrow unique(\{b_i = b_i \bigcup b_j \text{ if } b_i \bigcap b_j \neq \emptyset; \forall i, j\})$;
    /* Collapse the search space and update the
       building-block configuration according to
       the detected modules */
    $[M, V] \leftarrow UpdateModuleKnowledge(b)$;
```

### 4.3.2 *Noise Filtering*

Problems may arise if the inputs are too noisy; the local-search can not guarantee that modules will be expressed 100% in all cases. Accordingly, we must use an approach which enables noise filtering.

We assume that the local-search enables the correct expression of the module settings with a reasonably high probability i.e. this probability $p$ is larger then 0.75 in the case of each module. Then the following strategy can be applied: model building with SOM is employed to a small, randomly chosen subset of the total number of samples. The cardinality of the subset used in our experiments was set to 4. The higher is $p$, the slimmer is the probability that the subset formed by four converged states will contain more then one corrupted sample per block. Thus, if $p$ is high, the SOM will detect the dependent variables most of the time.

The SOM will also report false linkages more often, because the restricted number of four samples are not enough to correctly delimitate the modules with high probability. To alleviate this phenomena, we use the above described learning from randomly chosen subsets repeatedly $z$ times ($z = 32$ in this paper). We record in a square frequency matrix $F$ of length $|M(s)|$, the total number of times the SOM reported the $i^{th}$ and $j^{th}$ variable to belonging to the same module.

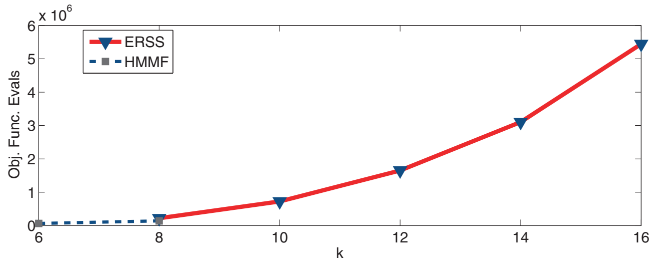

**Figure 3: Performance and scaling of the MBMM on the proposed test suites.**

$F$ will essentially be a dependency structure matrix (DSM), which is in turn an adjacency matrix representation of a graph where each entry $F_{ij}$ represents the dependency between node $i$ and node $j$ [22]. The closer is $F_{ij}$ to $z$ the higher is the interaction between node $i$ and node $j$.

To extract the information regarding module composition from $F$, we could use any DSM clustering technique which will find subsets of DSM elements so that nodes within a cluster are maximally interacting, and clusters are minimally interacting, like the one presented in [23].

However, if $p$ is high a more efficient linkage detection-filtering can be used. In each row $i$ of $F$, the numbers related to the truly dependent variables $j$ will be much higher then the rest of the entries. With a hardcoded threshold of $\epsilon_m \geq 0.75 \cdot z$ the delimitation of true dependencies from those induced by the noise can be done safely.

In the proposed method, using $\epsilon_m$ it is decided for each row $i$ from $F$, the set of dependent variables $j$ denoted by $\chi_i$ and a base module is formed from $b_i = \{i\} \bigcup \chi_i$. If no dependent variables are found in a row, $\chi_i = \emptyset$.

These base modules may be overlapping in some cases or even be subsets one of each other. Knowing that in the test functions of interest the modules are non overlapping ones, we use the circumstantial approach of merging together modules that contain at least one common element. In this way we enable the formation of larger modules and facilitate the faster decomposition of the search space.

The MBMM is summarized in Algorithm 1.

## 5. RESULTS

The performance of the proposed MBMM method was tested on the $ESRR$ function with even base module sizes from 8 to 16 and on $HMMD$ function of order 6 and 8. On the $ESRR$ function the parameters that reward certain block configurations were set to $r_1 = 1, r_2 = 2, r_3 = 4$. The $HMMD$ functions were used with $r_1 = 1$ and $r_2 = 2$. The $\zeta$ threshold for fitness variance in the model based macro-mutation was dynamically determined: using 100 trials, we computed the average fitness improvement $\delta_+$ of the successful mutations working on randomly generated states. A major fitness improvement was regarded as one above this value with more then 50%, thus $\zeta = 1.5 \cdot \delta_+$.

For test suites with block sizes up to 10, a total number of 100 independent runs were averaged. For problems with base module sizes of 12 the number of averaged runs was 30, respectively 10 for block sizes of 14 and 16.

The number of function evaluations used by an epoch of local-search was set to $5n^2$. The method showed a very good behavior, with 100% success rate on every test suite and with a very accurate and prompt model building. The converged solutions contained very little noise, showing that the biased mutation is effective in discovering modules and preserving them and thus the number of function evaluations required by Eq. 4 for an epoch of local-search is an overestimate for this case.

The performance and scalability of the method is depicted in Fig. 3. As k grows, random sampling required to discover blocks grows exponentially. Nevertheless, even for k as large as 16, the proposed method, using objective function guidance and biased search for module discovery, is able to find and combine all blocks accurately, within the bounds of $6e6$ objective function evaluations.

We repeated our experiments for the $ESRR$ test suite, up to block sizes of 14, using the simple, blind macro-mutation strategy, without the fitness variance analysis and preliminary model building. Here, whenever a new state is accepted, we only perform a simple greedy search upon this state, in order to discover better solutions, if any, in the close neighborhood of the new state.

The results are summarized in Table 1. The first column contains the basic module sizes. The scaling of function evaluations with regard to the block/problem size (remember that $n = k^2$), invested by the local-search in a single run in order to detect correct module settings, is presented in the $2^{nd}$ column. Column 3 contains the rate of success for each test suite, while the average total number of function evaluations until global optimum is reached, is reported in the last column.

$ESRR$ of size 8 and 10 is easily solved again in all cases, allocating the $5n^2$ function evaluations to the local-search. This is expected according to the scaling of $E$ vs. $5n^2$ depicted in Figure 1.

Equation 4 predicts the expected runtime for finding neutral blocks for module size 12, to be significantly bigger then $5n^2$ if we use only a blind mutation, which will destroy many times the already formed correct modules. This hypothesis held in our empirical findings also, as with $5n^2$ evaluations for the local-search, the MBMM did not perform reliably. Albeit it was able to converge to a global optima in some cases, the model building was poor due to heavy noise, taking many iterations to achieve the result, sometimes more then 10. To remediate this situation we doubled the value of $c_2$ from 5 to 10. This is still feasible as it respect our condition that $c_2 \leq k$. With the new setting for $c_2$ the method performed much better, always finding one of the optima.

For the hardest case tested here, with $k = 14$, even whit $c_2$ set to its maximum value of $14n^2$, is way less than required according to the scaling of $E$. Nevertheless, the solving of this particular test suite is also manageable, with blind mutation and relatively reasonable computational effort. We

| $k$ | LS - $c_2 n^{c_1}$ ($c_2 \leq k$) | Succ. rate | Avg. nr. obj. func. evals. |
|---|---|---|---|
| 8 | $5n^2$ | 100% | 3.442e5 |
| 10 | $5n^2$ | 100% | 1.425e6 |
| 12 | $10n^2$ | 100% | 7.547e6 |
| 14 | $5n^{2.39}$ | 100% | 2.655e7 |

**Table 1: Numerical results of the MBMM with the blind version macro-mutation on the $ESRR$.**

modify the Algorithm 1 in the following way: the limit of $c_2 n^2$ in the local-search is changed from number of function evaluations to *epochs*. This implies that $c_2 n^2$ mutations are performed and whenever the newly obtained solution is better or equal as the current one, a "free" greedy search is also used (the function evaluation consumed by the greedy search does not count, as we are running fixed number of epochs). Theoretically, if all mutations would result in at least equally fit solutions, the number of objective function evaluations would be proportional with $n^3$. Of course this is not the case; as modules are discovered, many blind mutations are detrimental ones, destroying the already formed blocks.

Working with the modified version of the MBMM that applies local-search for $5n^2$ *epochs*, the average number of function evaluations consumed by a single run of the local-search was less then $1.5e6$. Solving $5n^{c_1} - 1.5e6 = 0$ yields $c_1 \approx 2.39$. $5n^{2.39}$ is greater than the value of $E$ for $k = 14$, thus as expected the MBMM is able to discover and recombine the blocks of this problem successfully in all cases.

Due to the strong exploration performed by the biased model based macro-mutation, neutrality of $ESRR$ the massive multimodality and average case deceptiveness of the $HMMD$ function is overcame, as shown in the results. The method correctly identified one of the global optima and provided an exact model building in all runs. Even if the search space is extremely hard, due to the random shuffling and because of the astronomical number of aleatory placed local optima, the MBMM is able to quickly solve this problem by building and exploiting an accurate problem decomposition. When applying only blind macro-mutation, which often destroys already formed blocks, leading to wasted function evaluations, large neutral blocks can be still reliably detected by using in each local-search epoch a number of function evaluations as required by Eq. 4.

Note that the parameters of the method were not hand tuned. We worked with some initial estimates that turned out to work fine, thus there may be room for improvements. We especially suspect that reducing the number of samples from the memory to 12 will still provide good results with high probability, while the number of objective function evaluations used would drop by 25%. Furthermore, when using biased macro-mutation to protect already formed modules, for the problems under investigation we found that Eq. 4 is an overestimate, a value close to $m2^k$ would be more appropriate.

## 6. CONCLUSIONS

As they use only *random sampling* to supply initial building-blocks, classical PMBGAs are lower bounded in the population size by the exponential of the order of dependencies covered by the probabilistic model. For larger order of dependencies, the cost of model building from the exponentially growing populations, quickly exceeds feasible limits and uprise beyond economical practicality.

The paper propose a new paradigm and method (Model Based Macro-Mutation Hill-Climber), where initial building-blocks supply is achieved by employing strong exploration techniques, which nevertheless follow *objective function guidance* whenever available. The macro-mutation based search used, has the capability of sampling a wast portion of the search space, howbeit greatly differs from random sampling by being a hill-climber. It does a biased search by never accepting states with lower fitness. Furthermore, by analyzing objective function variance it can filter out detrimental mutations which destroy already formed blocks.

Investing the function evaluations into an exploratory local-search can facilitate the discovery of large modules. The method can also decide between the most fit module settings and their most competing schemata. Thus, the total number of samples needs to be just large enough, to make the delimitation of different modules possible with suitable machine learning techniques.

Empirical results confirm that the proposed method can solve with great reliability, very hard problems with huge neutral blocks or massively multimodal, deceptive problems efficiently, by using $c_2 n^2$ function evaluations in each local-search epoch, where $c_2$ is restricted to be smaller or equal to the base module sizes.

The macro-mutation based search can be hybridized with other probabilistic methods, where it is used only as a pre-processor to discover the elementary modules of a problem. Machine learning and further exploration would be performed by the host method.

## Acknowledgments

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

## 7. REFERENCES

[1] 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.

[2] D. E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[3] D. E. Goldberg, K. Deb, and J. Horn. Massive multimodality, deception, and genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Elsevier Science Publishers, B. V.

[4] D. E. Goldberg, K. Sastry, and T. Latoza. On the supply of building blocks. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 336–342, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[5] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999.

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

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

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

[9] Z. Li and E. D. Goodman. Learning building block structure from crossover failure. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1280–1287, New York, NY, USA, 2007. ACM.

[10] 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.

[11] S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, Dept. General Engineering, University of Illinois, Urbana, Illinois, 1995. Also, IlliGAL Report No. 95001.

[12] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In F. J. Varela and P. Bourgine, editors, *Proc. of the First European Conference on Artificial Life*, pages 245–254, Cambridge, MA, 1992. MIT Press.

[13] M. Mitchell and J. H. Holland. When will a genetic algorithm outperform hill climbing? In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 647–647, San Mateo, CA, USA, July 1993. Morgan Kaufmann.

[14] 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.

[15] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Linkage problem, distribution estimation, and bayesian networks. *Evolutionary Computation*, 8(3):311–340, 2000.

[16] M. Pelikan, K. Sastry, and D. E. Goldberg. Scalability of the bayesian optimization algorithm. *Int. J. Approx. Reasoning*, 31(3):221–258, 2002.

[17] 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.

[18] K. Sastry and D. E. Goldberg. Let's get ready to rumble: Crossover versus mutation head to head. In *GECCO '04*, pages 126–137. Springer, LNCS, vol. 3103, June 2004.

[19] D. Thierens. Scalability problems of simple genetic algorithms. *Evolutionary Computation*, 7(4):331–352, 1999.

[20] R. A. Watson, G. Hornby, and J. B. Pollack. Modeling building-block interdependency. In *Proc. of PPSN V*, pages 97–108, London, UK, 1998. Springer, LNCS.

[21] 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.

[22] A. Yassine, D. Falkenburg, and K. Chelst. Engineering design management: an information structure approach. *International Journal of Production Research*, 37(13):2957–2975, 1999.

[23] T.-L. Yu and D. E. Goldberg. Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In *GECCO '06:*, pages 1385–1392, NY, USA, 2006. ACM Press.