Transforming Evolutionary Search into Higher-Level Evolutionary Search by Capturing Problem Structure

Mills, Rob; Jansen, Thomas; Watson, Richard

Published in:
IEEE Transactions on Evolutionary Computation

DOI:
10.1109/TEVC.2014.2347702

Publication date:
2014

Citation for published version (APA):
Mills, R., Jansen, T., & Watson, R. (2014). Transforming Evolutionary Search into Higher-Level Evolutionary Search by Capturing Problem Structure. *IEEE Transactions on Evolutionary Computation*, *18*(5), 628-642. https://doi.org/10.1109/TEVC.2014.2347702

Transforming Evolutionary Search Into Higher-Level Evolutionary Search by Capturing Problem Structure

Rob Mills, Thomas Jansen, Richard A. Watson

Abstract—**The intuitive idea that good solutions to small problems can be reassembled into good solutions to larger problems is widely familiar in many fields including evolutionary computation. This idea has motivated the building-block hypothesis and model-building optimisation methods that aim to identify and exploit problem structure automatically. Recently, a small number of works make use of such ideas by learning problem structure and using this information in a particular manner: these works use the results of a simple search process in primitive units to identify structural correlations (such as modularity) in the problem that are then used to redefine the variational operators of the search process. This process is applied recursively such that search operates at successively higher scales of organisation, hence *multi-scale search*. Here we show for the first time that there is a simple class of (modular) problems that a multi-scale search algorithm can solve in polynomial time that requires super-polynomial time for other methods. We discuss strengths and limitations of the multi-scale search approach and point out how it can be developed further.**

Index Terms—**Evolutionary computation, automatic problem decomposition, linkage-learning, modularity, scalability**

I. Introduction

MULTI-SCALE search is an approach to evolutionary optimisation that involves the repeated creation of new variational units, each time transforming the search to a higher, more refined scale. The basic evolutionary tenet of selecting among high-fitness variants is maintained, but unlike other conventional approaches, variation is repeatedly re-scaled by canalising novel combinations of units discovered at the previous scale. Here we focus on identifying a niche for which multi-scale search is unique in scaling polynomially, and via this result, elucidate conceptual differences between this and existing approaches. We also point out limitations of the approach and discuss ways to develop it further.

A. Divide-and-conquer, and multi-scale search

Solving a problem by dividing it into smaller more manageable problems, or in a bottom-up process, assembling good solutions to small problems to find good solutions to large problems, is a familiar concept in conventional engineering (problem decomposition and modular design [1], [2], [4]) and artificial intelligence (*e. g.*, dynamic programming, memoisation and chunking [5]–[7]) as well as evolutionary computation [8]–[10]. The building-block hypothesis (BBH) [11], [12] in particular, suggests that the reason the genetic algorithm (GA) works well when it does is because crossover allows good low-order (small) building blocks to be recombined to find good building-blocks of higher order [13], [14]. The notion that this process scales up the search from exploring combinations of bits to exploring combinations of blocks, and so on, is quite clear in the original motivation [11], [12].

Here we concentrate on a general class of algorithms that we call *multi-scale search*. They have the property that the search operators are (repeatedly) redefined, creating a new search process at a higher scale of organisation. In particular, this redefinition is achieved by identifying strongly correlated subsets of variables from multiple results of local search in the current search space, and consequently reducing the dimensionality of subsequent search. Creating a macro-variation operator exploits these correlations by enabling specific simultaneous state changes in the problem variables *i. e.*, modular variation. Thus, a multi-scale search method initially searches combinations of the original problem variables, and as it progresses, changes the search space so that it starts to search in combinations of modules – and this transformation may be repeated at several scales of organisation. Here, unlike the action of crossover shown in [15], we are only interested in methods that are applicable to random linkage problems where there is no *a priori* information about the location or even the size of modules at any scale of organisation. Aside from our own work [16], [17] and Iclanzan and Dumitrescu's building-block hill climber (BBHC) [18], there are a small number of earlier works that also exhibit features of multi-scale search with different emphases including [19]–[21]. We discuss related work in Section II.

Our recent work has developed a number of methods and test cases that explore multi-scale search extensively. This includes the MACRO algorithm [16] which has several variants: MACRO-H (as used in most of this paper) makes explicit, discrete and irreversible 'joins' between problem variables ('H' for hard joins). MACRO-S, in contrast, makes probabilistic joins and modules are never encapsulated explicitly ('S' for soft joins). Using the latter we have shown that this type of method can work effectively in problems where the modules are not disjoint, which thus differ qualitatively from separable test problems. We have also shown that the algorithm can

R. Mills and R. A. Watson are members of ECS, University of Southampton, SO17 1BJ, UK. email: `rob.mills@soton.ac.uk`.

T. Jansen is a member of Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, UK

be distributed such that the join between any given pair of variables is the sole responsibility of the two variables involved, rather than the mandate of a centralised model-building process or a centralised data-structure [17]. This type of distributed model then becomes a type of adaptive network [22] where the correlation of variables affects the structure of joins, and concurrently, the structure of joins affects the correlation of variables.

Because there are several different approaches to the general idea of multi-scale search, here we describe the main characteristics of multi-scale search that are common to simple versions of MACRO and the BBHC:

1) **Evolutionary search with the current units.** At the core of multi-scale search is a simple process of variation and selection. This is, in a sense, a 'local' search process but crucially it must be able to operate in a search neighbourhood that is defined by a set of variational units that change over time – it searches in the space of single-unit changes, not merely single-bit changes, and these units become larger and larger as search progresses. To distinguish from non-rescaling local search, we refer to this simple search process as unit-exploiting search. In principle, unit-exploiting search does not have to literally be a hill-climber – it may be any search method that exploits the current level of units identified, *e. g.*, a population-based method. In MACRO-S higher-level units are probabilistic entities (built on-the-fly from linkage information as needed), but in BBHC and MACRO-H they are simply sets (schemata) of units from the previous level that are permanently joined together.
2) **Redefining variational units.** While the lowest scale of search unit is defined by the problem variables given, the higher scales of search units (which effect a lower-dimensional search space) are automatically discovered as the algorithm progresses. The lower-dimensional spaces are created by defining new variational units, which are based on structural linkage information that is learned from a high-fitness set of genotypes provided by search at the previous scale. This information may be based on bijective mappings as in the BBHC or on correlations [17], [23].
3) **Separation of timescales.** The set of configurations from one scale of search, used to identify the structure for the next scale of search, must reveal high-fitness combinations of units. We can achieve this by searching with the current units rapidly, and forming new units relatively slowly. Specifically, with sufficient separation in these timescales, each configuration used will be at or near a local optimum in the neighbourhood defined by the current units. Multi-scale search is thus produced by repeatedly applying the two phases of: i) search with the current variational units; and ii) identification of linkage to form new variational units. Or more generally, it can be produced by applying these two processes continuously at different timescales [17], [24].
4) **Diverse sampling.** Lastly, the set of configurations used to identify the structure for the next scale of search must avoid being overly converged on one particular way of achieving high-fitness. Accordingly, search with the current variational units is applied repeatedly from many different initial conditions. This search may either be done in parallel, as in the version of MACRO used here, or in series by regularly perturbing the state variables of the search process [17], [23], [25], [26].

B. What is multi-scale search good for?

These characteristics accommodate a large space of possible approaches to multi-scale search as explored in the algorithms examined above. However, in this paper our aim is to return to something very simple in order to assess whether the various methods previously investigated have a sound algorithmic basis and to formally characterise exactly what multi-scale search can do that other algorithms cannot. For this investigation we consider MACRO-H, as defined in Section IV.

In this paper we consider separable and hierarchical problems. We start with separable problems and present a simple test problem that is sufficient for our purposes: a two-scale building-block style problem (*i. e.*, bits into blocks, and blocks into the problem as a whole, without any further hierarchical levels), with separable and disjoint modules.

We previously used a tight-linkage problem of this type to demonstrate building-block recombination in the GA [15] (see also Section II), but here we use a random-linkage version of this problem that has not previously been shown to be solvable in polynomial time by any other algorithm. Our main result is to show for the first time that multi-scale search, specifically MACRO-H, solves random linkage problems of this class in polynomial time, specifically, $O(n \log n)$, where n is the problem size. We do this by showing that, unlike other algorithms, MACRO-H is polynomial in both the size and the number of modules of a separable problem. Moreover, a hierarchical version of this problem is not separable (it contains $O(n)$ dependencies). But because MACRO-H identifies modules at each level explicitly and in polynomial time, it can thus solve the next level in polynomial time, and so on.

The remainder of the paper is structured as follows. In the following section we discuss previous work. In Section III we define the class of problems we are interested in and define a specific test problem. Section IV contains a formal definition of MACRO-H. We analyse the performance of MACRO-H on our test problem and explain why other approaches are much less efficient in Section V. In Section VI we provide simulation results to reinforce the analytic results. We consider a hierarchical variant of the test problem in Section VII. Afterwards, in Section VIII, we discuss limitations and possible extensions of the approach. Section IX concludes the paper.

II. Previous Work

A. On learning linkage and module sizing

Early work attempting to demonstrate building-block recombination on an idealised test problem famously failed to show that a GA could solve something that a hill-climber could not. This failure was mostly due to the use of test problems that do not exhibit local optima that can trap a hill-climber.

In other words, although the problem was easy for the GA, it was also easy for the hill-climber [14], [27].

As a consequence of this failure, the idea that low-order fitness contributions should be deceptive [28], thus leading a hill-climber to become trapped on inferior local optima, became common place in GA test problems/analytical treatments. However, although the deception made problems sufficiently difficult for a hill-climber, it also made it difficult for the GA to find good building-blocks. Specifically, fully-deceptive sub-problems have the property that all local fitness gradients within the block lead search away from the best solutions. Such problems can therefore only be solved by search that is exponential in the size of the block – this is usually provided by making the population size sufficiently large such that solutions to building blocks are present in the initial population [29]. For such an approach to retain an overall time complexity that is a polynomial function of the total problem size, the size of building-blocks must thus be limited to a constant – hence the notion of 'order-k delineable' and 'order-k separable' functions [28], [30], [31]. But if building blocks are small, and the bits of a block have tight linkage (*i. e.*, the bits of a block are close together on the genome) such that they will survive together, and be selected together, through multiple crossover events [11], then the solutions to blocks can be found in polynomial time by macro-mutation hill-climbing [32].

There has been a great deal of research effort directed at methods to discover and exploit problem structure in problems with random linkage (*i. e.*, where tight linkage cannot be assumed). Estimation of distribution algorithms (EDAs) address this issue by using sophisticated machine learning techniques to model the epistatic dependencies in a problem [33]. These algorithms, including in particular, the Bayesian Optimisation Algorithm, BOA [34], have proved to be a very fruitful integration of population-based search and probabilistic model-building, yielding strong results on a variety of problem classes [35]–[37]. These works show strong performance of EDAs, in many cases empirically outperforming or being competitive with domain-specific techniques. However, theoretical work in this area retains the assumption that the search required scales exponentially with the size of the module (the number of bits or sub-modules it contains at any given scale of organisation) and therefore modules, at any level of hierarchy, must be small (with respect to the number of units from the level below). These methods also focus attention on the mechanisms to detect and represent problem structure and somewhat de-emphasise the idea of (recursively) scaling-up from searching combinations of bits to searching combinations of blocks.

Meanwhile, some work [15], [38] has subsequently shown that it is possible to defeat local search methods, including the macro-mutation hill-climber, without randomising the linkage of the problem. Rather, this work uses modules whose size scales as a function of overall problem size, not a constant. Each of these individual modules can, with reasonable probability, be solved in polynomial time because they are not fully deceptive. But because each module does have multiple local optima that may trap a hill-climber, a hill-climbing process will not be able to solve a problem that has many such modules in polynomial time [15]. This work is thus successful in demonstrating that there is a type of problem that a type of GA can solve in polynomial time that nonetheless requires super-polynomial time for hill-climbing methods or a GA without crossover. Specifically, a simple, tight-linkage, separable modular problem can be solved by a GA with crossover in polynomial time given sufficient population diversity. To be exact, the time required by a hill-climber is exponential in the size of the modules and exponential in the number of modules – but since the size of modules and number of modules are mutually constrained (*i. e.*, the overall problem size is the product of the two) the overall time complexity of the hill-climber is not 'strongly exponential' in total problem size, n, rather it is exponential in $\sqrt{n}$, or 'weakly exponential' in n [15], [39]. Note that [15] retains the idea of progressing from searching combinations of bits (to find good blocks) to searching combinations of blocks (to solve the complete problem) as per the BBH. In fact, this type of problem defeats the hill-climber precisely because it requires combinatorial search at two different scales (see also [38]). However, in demonstrating the original tight-linkage operation of crossover, this work makes no attempt to solve random linkage problems.

B. Scaling up with new variational units

There have been a number of works that attempt to exploit problem structure by explicitly grouping together multiple components into meta-level components. Grouping mechanisms are present in evolutionary algorithms (EAs) that employ chunking mechanisms hierarchically [40], including some that use tree-based genetic programming substrates [41]–[43]; as well as symbiosis-inspired algorithms (*e. g.*, [44]–[47]).

The BBHC [18] also embraces this idea but has some additional features that have produced impressive results. BBHC explicitly exploits the idea of scaling-up search from searching combinations of bits to combinations of small blocks to searching combinations of large blocks, and so on – and it operates on random linkage problems by learning problem structure bottom-up. It is therefore, in a sense, a type of model-building algorithm – but it uses learned information in a different manner from other model-building methods. Specifically, it uses the results of a simple search process (over combinations of the primitive units/problem variables) to identify structural correlations in the problem that are then used to redefine the variational operators of the search process. These new operators create a new search process at a higher scale of organisation. Clearly the basic intuition behind this work is closely aligned with those of the other works we have discussed, but the emphasis is different – specifically, the way in which learned information is represented is somewhat simple compared to BOA, for example, but the way in which the learned information is exploited, *i. e.*, by redefining the variational operator, is an innovation not seen in BOA.

The linkage-tree GA is a further interesting development that builds a model of linkage between variables, deferring modelling of the variable assignments to a population; and searches with a combination of steepest ascent hill-climbing and model-informed crossover [21].

We can broadly classify the existing multi-scale search algorithms over four dimensions: the frequency of model update;

the type of join made between units; the unit representation; and the use (or not) of centralised data structures. MACRO-H (examined in this paper) learns new structure in a batch mode, as do [18], [19] and MACRO-S [16]. Continuous learning is used in [17], [23]. MACRO-H, [18], and [19] make hard, irreversible joins between units; whereas [16], [17] reshape future variation via soft associations between units. In [23] hard joins are made once a soft threshold is exceeded. The primitive units in MACRO-H and [16] represent values of each variable, rather than whole variables as per [17]–[19], [23]. Finally, [17] provide a decentralised implementation in which changes to joins between units are the sole responsibility of the units involved; the rest use centralised processes and data.

Iclanzan and Dumitrescu [18] have already shown that this type of algorithm can solve some problems faster than BOA. For example, on the non-separable hierarchical problems HIFF, HXOR [14] and hTrap [34] they empirically found a time complexity of $O(n \log n)$ (where n is the problem size) for the BBHC. In comparison, BOA scales empirically as $O\big(n^{1.62} \log n\big)$, closely matching the analytical prediction $O\big(n^{1.55} \log n\big)$ [30]. However, it has not yet been shown that there is any type of problem that can be solved by the BBHC in polynomial time that requires super-polynomial time for other methods. We argue here that the fundamental reason that previous work fails to show such a distinction is because it inherits out-dated assumptions from prior work – specifically, the idea that modules (at each level of the modular structure) have to be small because finding solutions to them requires time exponential in their size. Our aim is not to criticise this prior work: rather, we aim to emphasise the potential of this type of algorithm. Specifically, by showing that when we properly understand what it can do that other algorithms cannot, we can demonstrate a polynomial versus super-polynomial distinction from other algorithms.

Some of our early work [44] was the first to solve random-linkage hierarchical building-block problems and, correspondingly, the first to use an explicit redefinition of the search space through multiple scales by 'joining' the original problem variables together into larger and larger groups. Later work explored simplifications to the methods used to identify modules [47] and, as we gained a better understanding of its capabilities, we also simplified the type of test problem that exemplified the capabilities of this method [48]. The BBHC would later identify the same modules as those that were found by these algorithms, and exploit the learned modules in essentially the same way, *i. e.*, redefining the variational operator by joining variables together, but the BBHC achieved this redefinition in a more elegant and much less computationally expensive manner. In hindsight, we now understand that our earlier work [44], [47], like many EDAs, did not properly exploit search at lower levels of organisation to more effectively identify dependencies that guide the next level of organisation – a feature which is very clear in the BBHC and our later work [16], [17], [49] as analysed in this paper.

III. Problem Classes and a Test Problem

We concentrate on problems that have identifiable modules that contribute additively to the function value. Such problems have been considered in various contexts for different purposes. Jansen and Wiegand [50] call them (m, k)-separable.[1] We make use of their definition.

Definition 1. *A function* $f\colon \{0,1\}^n \to \mathbb{R}$ *is called* (m,k)-separable*, where* $m, k \in \{1, 2, \ldots, n\}$*, if there exists a partition of* $\{1, \ldots, n\}$ *into* m *disjoint sets* $I_1, \ldots, I_m$*, and if there exist a matching number of pseudo-Boolean functions* $g_1, \ldots, g_m$ *with* $g_j\colon \{0,1\}^{|I_j|} \to \mathbb{R}$ *such that*

$$f(x) = \sum_{j=1}^{m} g_j\left(x_{i_{j,1}} x_{i_{j,2}} \cdots x_{i_{j,|I_j|}}\right)$$

holds for all $x = x_1 \ldots x_n \in \{0,1\}^n$*,* $I_j = \left\{i_{j,1}, \ldots, i_{j,|I_j|}\right\}$ *and* $|I_j| \le k$ *for all* $j \in \{1, \ldots, m\}$*.*

We say f *is* exactly (m,k)-separable *if* f *is* (m,k)*-separable but not* (m', k')*-separable for any* $m' > m$ *or* $k' < k$*.*

Obviously, each function is $(1, n)$-separable. Here we address exactly $(n/k, k)$-separable functions for $k < n$. The modularity in such problems should, in principle, be exploitable if identified correctly and the strict separability better enables rigorous analytical results to be obtained; accordingly, this form of function has been used extensively in the literature (*e. g.*, [28], [30], [58]). One popular way of constructing such functions is to take an inseparable function (technically speaking, an exactly $(1, n)$-separable function) and concatenate n/k copies of this function defined over k bits each. We follow this approach and use TwoMax as the inseparable function. TwoMax is an almost symmetrical function with one global and one local optimum, one the all-ones bits string, the other the all-zeros bit string. It was introduced by Pelikan and Goldberg [51] and has been studied in different contexts (*e. g.*, [52], [53]). We use *scalable building-block* problem (SBB) as a name for the concatenated version of TwoMax.

Definition 2. *The function* TwoMax$\colon \{0,1\}^n \to \mathbb{N}$ *is defined as*

$$\textsc{TwoMax}(x) = \max\left\{\sum_{i=1}^{n} x[i], n - \sum_{i=1}^{n} x[i]\right\} + c\prod_{i=1}^{n} x[i]$$

for all $x \in \{0,1\}^n$ *and* $c \ge 0 \in \mathbb{N}$*.*

Let $n, k \in \mathbb{N}$ *with* $k > 1$ *and* $(n/k) \in \mathbb{N}$*. The function* SBB$\colon \{0,1\}^n \to \mathbb{N}$ *is defined as*

$$\textsc{sbb}(x) = \sum_{i=1}^{n/k} \textsc{TwoMax}\left(x^{(i)}\right)$$

for all $x = x[1]x[2] \cdots x[n] \in \{0,1\}^n$ *where* $x^{(i)}$ *is the* i^{th} *part of length* k *in* x*,* $x^{(i)} = x[(i-1)k+1]x[(i-1)k+2] \cdots x[ik]$*.*

We refer to the variable at locus i as $x[i]$ and variables at multiple loci as the concatenation, e. g., $x[i]x[j]x[z]$ for loci i, j, z. We use the usual notation for the concatenation of letters in a string, i. e., $b^k = \underbrace{bb\cdots b}_{k \text{ times}}$. So, for example, $0^3 10^2 = 000100$. Note that b^0 represents the empty string.

[1] [50] originally used the symbols r and s but here we use m and k as common in literature concerning building-block problems.

Definition 2 includes a parameterised bonus, c, for finding the 1^n solution. The primary focus of this paper uses nonzero c ($c=1$ where a numerical value is required), but in a later section we consider a hierarchical problem with TWOMAX as a basis function which does not distinguish between the two optima (*i. e.*, $c=0$). The specific value of the bonus is not critical, but it is important that $c > 0$ for the SBB, otherwise all local optima would in fact be globally optimal.

Since our aim is to compare the performance of MACRO-H with other randomised search heuristics we do not only consider SBB but a class of SBB-like functions where MACRO-H has the same performance. This perspective is adopted from black-box complexity [54] and we also consider the worst case performance taken over the whole class of problems as the measure of performance. We adopt the notions of Droste et al. [55] that coincide with the notions of unbiased black-box complexity [56].

Definition 3. *Let* $f\colon \{0,1\}^n \to \mathbb{R}$. *For any* $a \in \{0,1\}^n$ *we define* $f_a\colon \{0,1\}^n \to \mathbb{R}$ *by* $f_a(x) = f(x \oplus a)$. *For any permutation* π *over* $\{1,2,\ldots,n\}$ *we define* $f_\pi\colon \{0,1\}^n \to \mathbb{R}$ *by* $f_\pi(x) = f\left(x[\pi(1)]x[\pi(2)]\cdots x[\pi(n)]\right)$. *Moreover, we define* $f_{a,\pi} = (f_a)_\pi$.

Let $\mathcal{F} = \{f\colon \{0,1\}^n \to \mathbb{R}\}$ *be a class of functions. We define* $\mathcal{F}^* = \big\{f_{a,\pi} \mid f \in \mathcal{F},\ a \in \{0,1\}^n,\ \pi$ *permutation over* $\{1, 2, \ldots, n\}\big\}$.

For $f\colon \{0,1\}^n \to \mathbb{R}$ *we define* $f^* := \{f\}^*$.

It is easy to see that any search heuristic that treats 0-bits and 1-bits completely symmetrically as well as all bit positions completely symmetrically will have identical performance on each function $g \in f^*$ for any function f. Such search heuristics are called unbiased [56] and they are more reasonable than biased search heuristics as long as a specific search bias is not justified by problem-specific knowledge. Most randomised search heuristics are unbiased in this sense: this includes all mutation-based evolutionary algorithms and evolutionary algorithms with uniform crossover. A counterexample are evolutionary algorithms with 1-point or 2-point crossover because they are sensitive with respect to reordering of bits, *i. e.*, with respect to going from f to f_π.

Many search heuristic are also invariant with respect to arbitrary rescaling of function values (and MACRO-H is, too). This includes all rank-based evolutionary algorithms but excludes, for instance, evolutionary algorithms with fitness-proportional selection. While it would not change our results we will not consider this kind of unbiasedness here. Thus, in the following we concentrate on SBB*, the general class of SBB problems under Definition 3.

IV. MACRO-H

In this section we define a basic multi-scale search algorithm called MACRO-H. First we provide an informal sketch of the algorithm, which is followed by a description of notation, and then formal definitions in Algorithms 1 and 2.

Informally, the main action of MACRO-H occurs in two repeating phases: a search phase that exploits the current search units to find good solutions, and a structural identification phase that identifies new search units from these solutions. The next search phase uses these new search units to find better solutions, which then enables the discovery of better search units in the next structure identification phase, and so on. We can summarise the process as follows:

Initialise unit set V with all primitive units (*i. e.*, one unit for each value of each problem variable);

While computational budget remains, repeat:

1) **Perform unit-exploiting search multiple times** using the current variational units, V, to find several different local optima ('locally optimal' with respect to the current variational units);
2) **Update co-occurrence measure**, A, based on correlations between units present in these optima;
3) **Update unit set** V by combining the most strongly correlated units into composite units.

MACRO-H is defined for optimising a function $f\colon \{0,1\}^n \to \mathbb{R}$. One crucial notation for the algorithm's search process is that of a unit. A unit can be described as a partial assignment of values to the n bits of a search point $x = x[1]x[2]\cdots x[n] \in \{0,1\}^n$. We use the following notation for units. A unit is given as a pair (l, a) where $l \in \{0,1\}^n$ and $a \in \{0,1\}^n$. We use $l[i] = 1$ to indicate that the i^{th} variable is a member of this unit. If it is not we have $l[i] = 0$. The other part, a, is an assignment of the variables of the unit. The values $a[j]$ for those j with $l[j] = 0$ are not important and can be set arbitrarily. We say that $y \in \{0,1\}^n$ agrees with (l, a) if $l[i] = 1 \Rightarrow (y[i] = a[i])$ for all $i \in \{1, 2, \ldots, n\}$.

One important element of MACRO-H is the unit-exploiting search that can operate with both rescaled and primitive units. We define MACRO-H in the following as Algorithm 1 and define the unit-exploiting search separately as Algorithm 2.

The following examples illustrate the (membership, assignment) representation for an 8-bit problem, assigning between one and three values. Here we write $a[i] = *$ where the unit does not specify the value for a locus i (*i. e.*, $l[i] = 0$).

$$
\begin{aligned}
u_1 &= (\texttt{00100000},\ \texttt{**0*****})\\
u_2 &= (\texttt{00010100},\ \texttt{***1*0**})\\
u_3 &= (\texttt{01000000},\ \texttt{*1******})\\
u_4 &= (\texttt{11000010},\ \texttt{00****1*})\\
u_5 &= (\texttt{00101000},\ \texttt{**1*1***})\\
u_6 &= (\texttt{00000001},\ \texttt{*******1})
\end{aligned}
$$

The group construction process creates non-trivial units (Algorithm 1, lines 17–23). This process makes use of the pairwise associations between units recorded in the association matrix, A. These association strengths are computed as a function of the co-occurrence of the current units among the optima found in each round (lines 7–12). MACRO-H makes hard joins between units and thus uses $A_{u,v} \in \{0,1\}\ \forall u, v$. If we seed this process with u_1 and u_4, and $A_{1,2} = 1$, $A_{1,3} = 1$, $A_{1,v} = 0$, $\forall v \neq [2,3]$ and $A_{4,6} = 1$, $A_{4,v} = 0$, $\forall v \neq 6$, it will produce the following composite units:

$$
\begin{aligned}
u_1^+ &= (\texttt{01110100},\ \texttt{*101*0**})\\
u_4^+ &= (\texttt{11000011},\ \texttt{00****11})
\end{aligned}
$$

Algorithm 1: Algorithm MACRO-H

```
Input : Objective function $f\colon \{0,1\}^n \to \mathbb{R}$; total budget
        $b \in \mathbb{N}$; local search budget $b_{\text{search}} \in \mathbb{N}$; number
        of 'parallel' local searches $d \in \mathbb{N}$
Output: Best search point found
// Initialise unit set $V$ with all
   primitive units
1  $V := \{ (0^{i-1}10^{n-i}, 0^n), (0^{i-1}10^{n-i}, 1^n)$
2     $\mid i \in \{1,\dots,n\}\}$;
3  while computational budget $b > 0$ :
       // Perform 'local' search $d$ times;
          can be done in parallel
4      for all $i \in \{1,\dots,d\}$ :
5          $y_i :=$ UE-SEARCH$(f, b_{\text{search}}, V)$; // Returns
               $y_i \in \{0,1\}^n$
6          $b := b - b_{\text{search}}$;
       // Update co-occurrence measure $A_{u,v}$
7      for all $u, v \in V$ :
8          if $\{i \in \{1,\dots,d\} \mid y_i$ agrees with $u$ or $v\} \neq \emptyset$ :
9              $c := \frac{|\{i\in\{1,\dots,d\} \mid y_i \text{ agrees with } u \text{ and } v\}|}{|\{i\in\{1,\dots,d\} \mid y_i \text{ agrees with } u \text{ or } v\}|}$;
10         else:
11             $c :=$ undefined;
12         $A_{u,v} := \begin{cases} 1 & \text{if } c = 1, \\ 0 & \text{otherwise;} \end{cases}$
       // Update unit set $V$
13     $V' := \emptyset$;
14     for all $u \in V$ :
15         Remove $u$ from $V$;
16         if $\exists y \in \{y_1, y_2, \dots, y_d\}\colon y$ agrees with $u$ :
               // construct group $(l, a)$
17             $(l, a) := i$;
18             for all $v = (l', a') \in V$ in random order :
19                 if $A_{u,v} = 1$ :
20                     for all $i \in \{1,\dots,n\}$ with $l'[i] = 1$
                       and $l[i] = 0$ :
21                         $l[i] := 1$;
22                         $a[i] := a'[i]$;
23                     Remove $v$ from $V$;
24             $V' := V' \cup \{(l, a)\}$;
25     $V := V'$;
26 return some $y_i$ with $i \in \{1,\dots,d\}$ and
   $f(y_i) = \max\{f(y_1), f(y_2), \dots, f(y_d)\}$
```

each specifying values at four loci in this case. Different units participate in different groupings; and it may be the case that not all units are used at all. Note that although the association matrix entries are all pairwise, any number of constituent units can form a new group; and that there is no requirement for the constituent units to be contiguous, nor of the same order.

The capacity to recurse is integral to MACRO-H, and later in the paper we will illustrate this ability with a hierarchical test problem (Section VII). However, we start by analysing the runtime scaling on separable (non-hierarchical) problems.

Algorithm 2: Algorithm UE-SEARCH

```
Input : Objective function $f\colon \{0,1\}^n \to \mathbb{R}$; local search
        budget $b_{\text{search}} \in \mathbb{N}$; set of search units $V$
Output: Best search point $y$ found after $b_{\text{search}}$ steps of
        local search
// Construct initial point $x$ randomly
   based on $V$
1  $y := 0^n$;
2  while $\exists i \in \{1,\dots,n\}\colon y[i] = 0$ :
3      for all $(l, a) \in V$ in random order :
4          for all $i \in \{1,\dots,n\}$ :
5              if $y[i] = 0$ and $l[i] = 1$ :
6                  $y[i] := 1$;
7                  $x[i] := a[i]$;
8  $t := 1$;
9  Fix a random order, $\rho$, on $V$;
10 while $t < b_{search}$ :
11     $t := t + 1$;
       // Create $y$ as random 'neighbour' of
          $x$
12     $y := x$;
13     while $y = x$ :
14         Let $(l, a) \in V$ be next element in order $\rho$; if $\rho$ is
           exhausted then set $\rho$ to a new random order on $V$;
15         for all $i \in \{1,\dots,n\}$ :
16             if $l[i] = 1$ :
17                 $y[i] := a[i]$;
       // Select better search point
18     if $f(y) \geq f(x)$ :
19         $x := y$
20 return $x$
```

V. Analysis on the Test Problem

We analyse the performance of MACRO-H on SBB* and follow the common practice of using the number of function evaluations as a measure of time. Since a crucial component of MACRO-H is local search (at least in the first iteration) there is always a chance that MACRO-H gets stuck in a local optimum if the objective function is not unimodal. This implies that the most frequently used performance measure, the expected optimisation time, is not useful. One possible alternative is to analyse the time until an optimum is found with high probability. Another alternative would be to consider some kind of restart strategy. However, restarts add another layer of complexity, and results about restarts are easy to obtain from the first kind of results. Thus, we concentrate on the time that is sufficient to find a global optimum with high probability. Wegener [57] uses this fact to formally define efficiency measures in this spirit.

Theorem 1. *Let $n, k \in \mathbb{N}$ be given with $k > 1$ and $(n/k) \in \mathbb{N}$. Let SBB* be defined over n bits with n/k parts of equal length*

k. MACRO-H *finds a global optimum on any function $f \in$ SBB* in $O(n + n\log(n/k))$ function evaluations with probability $1 - O(1/n)$ if the individual search budgets are $\Theta(n)$ and large enough and $d = \Theta(\log n)$ and $d \geq 10.43\ln(n)$ holds.*

Proof. The specific local search implementation in UE-Search (Algorithm 2) guarantees that the time needed to find a local optimum in each of the components is $O(n)$. Since we assume the individual search budgets to be large enough the local optima are correctly located. This is due to the fact that all $f \in$ SBB* are linearly separable and the specific implementation of local search guarantees that all units are considered before any unit is considered a second time. This implies that the total number of improving steps (i. e., the sum of improving steps over all parts) is bounded by $O(n)$. We assume that the search budget is sufficiently large to allow for this to happen. Thus, at the end of the local search we have either the local or the global optimum in each of the n/k parts. Note that the two optima in each part each are found with equal probability $1/2$.

If for each pair of parts we have at least one run of UE-Search where in both parts the global optimum is found we know that (1) in each part the global optimum is identified as a component (not necessarily on its own) and (2) that if the global optimum appears only as part of a larger component all parts of this larger component correspond to global optima. This implies that the next round of MACRO-H that operates on these components will identify the global optimum of SBB*.

For a fixed pair of parts the global optima are found in both parts by one run of UE-Search with probability $1/4$. The probability that this does not happen in all d runs is bounded above by $(3/4)^d$. Thus, the probability to have one of the $\binom{n}{2}$ pairs where this is the case is bounded above by $\binom{n}{2} \cdot (3/4)^d < n^2 \cdot (3/4)^d$. With $d \geq 10.43\ln(n)$ we have $n^2 \cdot (3/4)^d < 1/n$ so that the probability is bounded by $O(1/n)$. $\square$

The proof of Theorem 1 can easily be generalised to similar classes of functions. We state the more general result as a corollary.

Corollary 1. *Let $n, k \in \mathbb{N}$ be given with $(n/k) \in \mathbb{N}$. Let $f\colon \{0,1\}^n \to \mathbb{R}$ be a $(n/k, k)$-separable function such that the following conditions hold.*

1) *Each sub-function has at most two local optima (at least one of which is a global optimum, of course).*
2) *In each sub-function local search as implemented in Algorithm 2 finds the global optimum with probability at least $p(n)$ if the search budget is sufficiently large.*
3) *In the first round of* MACRO-H*, Algorithm 2 finds a local or global optimum in each sub-function in at most $t(n)$ steps.*

MACRO-H *finds a global optimum of f within $O\big(p(n)^{-2}t(n)\ln n\big)$ function evaluations with probability $1 - O(1/n)$ if the local search budgets are $O(t(n))$ and are sufficiently large and $d = \Theta\big(\ln(n)/p(n)^2\big)$ and $d \geq 3\ln(n)/p(n)^2$ holds.*

Proof. In comparison with the proof of Theorem 1 the local search budget and d are changed. Moreover we do not assume that the local optima for one sub-function are necessarily bit-wise complements of each other. This generalisation may lead to incorrect recognition of larger units including bits from two different local optima that coincide. Since the number of local optima is restricted to 2 this generalisation cannot cause any problems and the global optimum will still be found no later than in the second round of local search.

We consider a pair of components, and see that the global optima are found in both components within one run of UE-Search with probability at least $p(n)^2$. Thus, the probability that this is not the case in all of at least $3\ln(n)/p(n)^2$ runs is bounded above by $(1 - p(n)^2)^{3\ln(n)/p(n)^2} \leq e^{-3\ln n} = 1/n^3$. The probability to have this in at least one of $\binom{n}{2}$ pairs is bounded above by $\binom{n}{2} \cdot (1/n^3) = O(1/n)$. $\square$

One extreme application of Corollary 1 is using the well known trap function (see, *e. g.*, [28], [58]) as the sub-function to create an exactly $(n/k, k)$-separable function. Since the probability to find the global optimum for a trap function as sub-function is $\Theta\big(2^{-k}\big)$ we obtain $O\big(2^{2k} n \log n\big)$ as upper bound on the optimisation time of an appropriately parameterised MACRO-H. Note that this bound is polynomial in n as long as $k = O(\log n)$ holds.

Theorem 1 states that MACRO-H is efficient on every instance of SBB* for any combination of n and k (given that the parameters are set appropriately). We now compare this performance with other randomised search heuristics to clarify the specific strength of MACRO-H. In order to do this we concentrate on the case $k = \sqrt{n}$ (and assume that $\sqrt{n} \in \mathbb{N}$ for convenience). Note that since we discuss large classes of algorithms in the following we provide key ideas for rigorous proofs and not proofs themselves. It is not too difficult to create concrete proofs for concrete search heuristics.

Local search with restarts cannot find the optimum of any SBB problem of this kind in $o\Big(2^{\sqrt{n}}\Big)$ steps. Local search finds either a local or a global optimum in each of the $\sqrt{n}$ parts with equal probability, independently in each part. Thus, the probability to find the global optimum simultaneously in all parts is $2^{-\sqrt{n}}$.

Mutation-based evolutionary algorithms are in a situation similar to local search but can escape local optima with large mutations. Since such a mutation requires the flipping of $\Omega(\sqrt{n})$ bits simultaneously such a mutation has probability $2^{-\Omega(\sqrt{n})}$.

Evolutionary algorithms with crossover are also not better off. Uniform crossover also allows the exchange of parts which are only locally optimal with probability $2^{-\sqrt{n}}$. The use of k-point crossover cannot help since in SBB* we have functions where the bits of the parts are not contiguous but distributed over the whole bit string.

Estimation of distribution algorithms have the problem of requiring a population size $\Omega\Big(2^{\sqrt{n}}\Big)$ and will therefore also be very inefficient [29], [59], [60].

A proof for polynomial time complexity of the BBHC on this problem class has not been provided by the literature on building-block hill-climbing, but we suspect that it could yield a similar complexity to that for MACRO, since we consider both to be instances of multi-scale search.

For MACRO-H measuring the number of fitness evaluations neglects the cost of updating the unit test set V. However, for most objective functions this is not a significant contribution (at least in asymptotic notation). The update of the unit test set (see lines 13–25 in Algorithm 1) has run time $O(n^2 d)$ which can be seen as follows. The outer loop over V is carried out $O(n)$ times. The test in line 16 can be carried out in time $O(nd)$ so it contributes a total of $O(n^2 d)$ to the run time. The more costly inner loop (lines 18–24) has cost $O(n^2)$. Since it is only carried out if the test in line 16 is positive this can happen at most d times so that this loop also contributes $O(n^2 d)$ summing up to $O(n^2 d)$. To obtain the cost for the function evaluations it is safe to assume that most objective functions require time $\Omega(n)$ to be evaluated since in order to compute the function value it is often necessary to consider every bit or at least a constant fraction of them. In each UE-SEARCH we usually perform $\Theta(n)$ function evaluations. The number of times we call UE-SEARCH is d so that we have in total a computational effort of $\Omega(n^2 d)$ due to function evaluations. This cost is not smaller than the computational effort for updating the unit test set V.

EDAs also incur a comparable level of computational costs beyond function evaluations. For instance, the search for Bayesian network structure and parameters in BOA costs $O(n^2 N + n^3 k)$, and sampling costs $O(nkN)$, for a population size of N [62]. MK-EDA [37] only fits parameters to a dependency structure that is given *a priori* (rather than learning the problem structure), which costs $O((n-k)N)$, and Gibbs sampling costs $O(n 2^k N)$. The tree EDA model in [37] costs $O(n^2 N)$ to train and $O(nN)$ to sample.

VI. SIMULATED EXPERIMENTS

In this section we confirm the analytic results from above empirically, by applying MACRO-H to three exactly $(n/k, k)$-separable functions: 1) SBB; 2) concatenated trap functions; and 3) a generalisation of SBB that relaxes some properties.

A. *Simulation on the* SBB* *problem class*

As discussed above, the SBB problem causes difficulty for local search when it has many sub-problems, and is difficult for mutation-based escape from a local optimum when the sub-problems are large. Here we set $k = \sqrt{n}$ to balance these two factors; although Theorem 1 holds for any setting of k. We use Theorem 1 to define the value for the one free parameter in MACRO-H, d. Specifically, $d = \lceil 10.43 \ln(n) \rceil$, which gives $d \in [34, 68]$ for the problem sizes $n \in [25, 625]$ tested. In addition, we report the performance of MACRO-H with the unit transformation mechanism disabled (by skipping lines 7–25 of Algorithm 1). This variant algorithm behaves indistinctly from a restart hill-climber, and is predicted to scale as $o\left(2^{\sqrt{n}}\right)$ as noted above. While empirical data can support successful results by example, demonstrating a negative result cannot support the general case since such results are susceptible to poor parameter tuning. Thus to provide some intuition for the scalability of EDAs, we also plot curves corresponding to the theoretical population sizing requirements based on the gambler's ruin model [59], [62] and the refined theory for entropy-based models [60]. Figure 1 shows the mean number of evaluations needed to reach the global optimum, which confirms that the runtime of MACRO-H scales as expected, namely, $O(n \log(n/k)) = O(n \log \sqrt{n}) = O(n \log n)$.

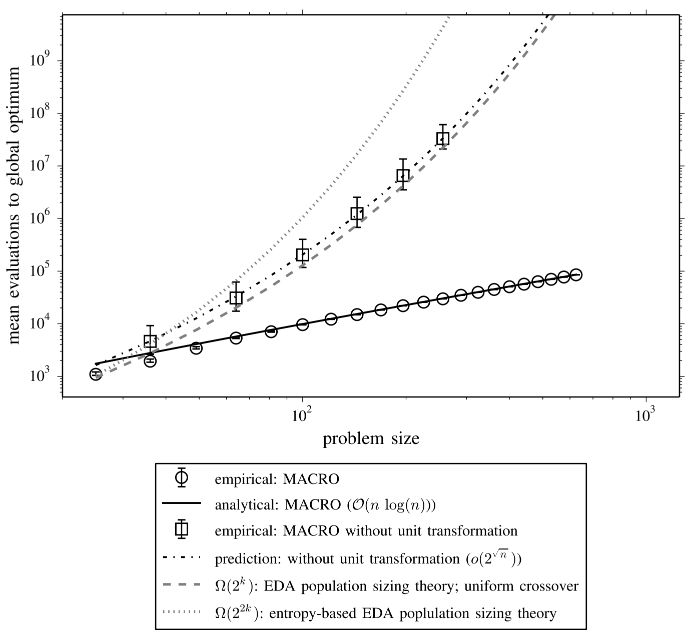

Fig. 1: Scaling for various algorithms on the SBB problem. The dashed and dotted lines illustrate theoretical lower bounds reflecting a factor that is exponential in the size of blocks. Circles are empirical results for MACRO; we also test a variant algorithm with unit transformation disallowed (squares), indicating how crucial transformation is for efficient operation. Error bars are shown but for MACRO are negligible with respect to the size of markers. Both algorithms find the global optimum in all 100 independent repeats. Solid and dot-dashed lines are analytic predictions of runtimes for these algorithms. These results show that MACRO-H solves the SBB problem in polynomial time, in very close agreement with analysis.

B. *Concatenated trap functions*

Following the application of Corollary 1, we empirically validate MACRO-H on the concatenated trap function. Here we set $k = 5$, and thus Figure 2 shows the runtime of MACRO-H to scale as $O(2^{2k} n \ln n) = O(n \log n)$ as expected. We set $d = \lceil 64 \ln(n) \rceil$ which has the asymptotic complexity as predicted by theory but is numerically smaller; empirically, MACRO-H succeeds in 100 of 100 repeats with this value for d, providing some indication of how tight the upper bound is (for reference, Figure 2 shows the analytical upper bound). We also plot results from the literature on this problem: BOA taken from [61]; hBOA from [62]; MOA from [63]. To our knowledge, these are the only published runtime results of EDAs on concatenated 5-traps. The results collected in this figure confirm that MACRO-H is able to solve the problem in sub-quadratic time, along with several other methods that learn the structure. The exact performance of these different algorithms varies somewhat, but all scale sub-quadratically in the problem size. Hence, this problem does not qualitatively discriminate between EDAs and multi-scale search approaches.

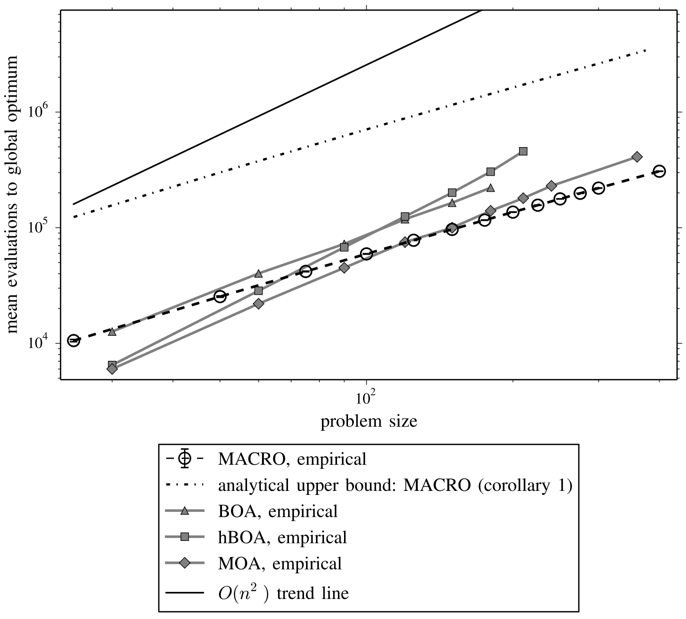

Fig. 2: Runtimes on the concatenated trap-5 problem. Dot-dashed line: MACRO upper bound of $O(n \log n)$, from theory in Section V. Circles and dashed line are empirical results using $d = \log(n)/4p(n)^2$, successful in all 100 independent repeats. Grey lines: empirical results for various other algorithms, taken from literature (see text for data sources). The solid black line indicates an $O(n^2)$ trend. MACRO-H grows sub-quadratically in n, but trap-5 problems are not sufficiently difficult to distinguish MACRO-H from other algorithms.

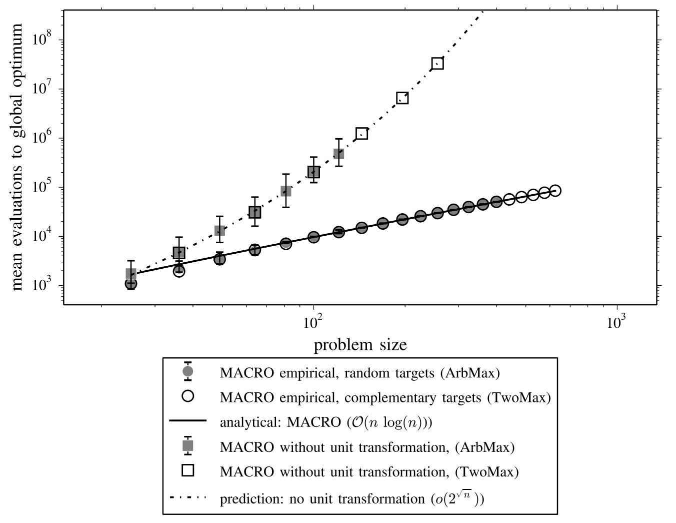

Fig. 3: Runtime scaling results on the SBB$_A$ problem, for MACRO-H and for the variant with unit transformation disabled (filled symbols). Both algorithms find the global optimum in all 30 independent repeats. For comparison we also replicate the data for the SBB problem from Fig. 1 (open symbols, error bars omitted), showing a strong agreement with the analytical expectations. Importantly, MACRO-H is able to discover which units should associate without the assumption that the complementary units should also associate to scale as $O(n \log n)$.

C. Relaxing complementarity in the SBB* problem class

Above we showed that MACRO-H is capable of efficiently solving the SBB* problem class in Theorem 1. Here we examine how MACRO-H performs on a variant of the SBB problem that does not use complementary optima in each block, confirming the expanded result from Corollary 1.

Definition 4. *The function* ARBMAX: $\{0,1\}^n \to \mathbb{N}$ *is defined as*

$$\textsc{ArbMax}(x) = \max\{n - \|t_1 - x\|, n - \|t_2 - x\|\} + \prod_{i=1}^{n}(t_1[i] \Leftrightarrow x[i])$$

for all $x \in \{0,1\}^n$*, where* $\|a - b\|$ *is the Hamming distance between patterns* a *and* b*, and where* $t_1, t_2 \in \{0,1\}^n$ *are target bitstrings with arbitrary positions, s.t.* $\|t_1 - t_2\| > 1$.

We create SBB$_A$ by concatenating n/k k-bit ARBMAX subfunctions (just as Definition 2 uses TWOMAX subfunctions in SBB). SBB$_A$ is very similar to SBB, except that the targets are no longer required to be at 0^n and 1^n; and crucially, are not necessarily full bit-complements of one another.

For this set of experiments, each variable is set uniformly at random in t_1 and t_2, independently for each block (and different in each repeat). Figure 3 reports the mean time to find the global optimum, showing that MACRO-H scales just as efficiently on this problem with random targets as it does in the more special case of complementary targets.

Why does this problem not affect the performance? MACRO-H learns which bit-values co-occur rather than which bits are co-extensive. Accordingly, MACRO-H can form useful associations between bit-values without assuming that the complementary bit-values are associated, or more generally, without making any assumption that the complement of fit schema will also be fit.

Figure 4 shows some example targets and the variational units that MACRO-H finds after the first iteration. The composite units created in response to target set a) match exactly those targets – and thus in subsequent unit-exploiting search, introducing one unit will replace variables for an entire module as described. In b) and c), the composite units created correspond not to an entire module, but to the variables at which the two targets mismatch, as expected. Any other variables necessarily match in both targets, and thus are not joined to either one of the two composites for that module (or indeed beyond that module). However, for these loci where targets match, subsequent unit-exploiting search trivially discovers the correct value. Because the composite units correspond to all mismatching variables across targets in a module, introducing a composite is sufficient to move between optima. This behaviour is exactly what we see when all variables in a module mismatch across targets. Therefore, MACRO-H discovers associations in the first iteration that are entirely sufficient to solve SBB$_A$ in the second iteration.

Figure 5 further supports these examples by showing that the number of blocks solved by MACRO-H is not affected by

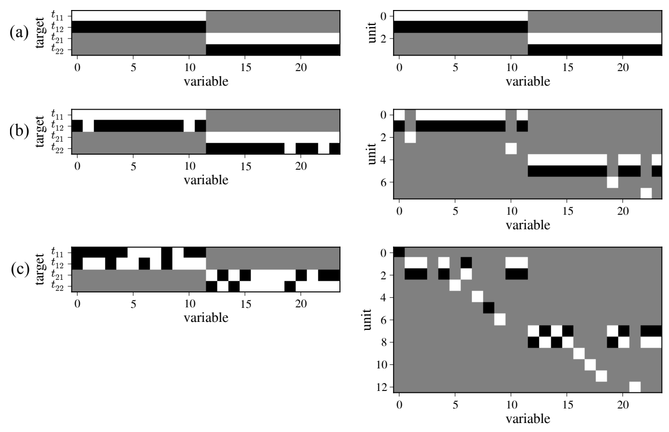

Fig. 4: Targets (left), and variational units that MACRO forms after one iteration (right), for different forms of the SBB/SBB$_A$ problem. Data is from a single run with $k = 12$, $m = 2$; other repeats are characteristically similar. (a) complementary optima, here chosen as $t_{i1} = 1^k$, $t_{i2} = 0^k$ for straightforward visualisation; (b) non-complementary targets, using a specific case where $t_{i1} = 1^k$, t_{i2} is two random bits away from 0^k; (c) two random targets. Black/white corresponds to 0/1, while any undefined values are in grey. Targets specify a value for all variables in one module. The units that are formed, ranging in order from 1 to 12, are sufficient for subsequent iteration of unit-exploiting search to find the global optimum in all cases.

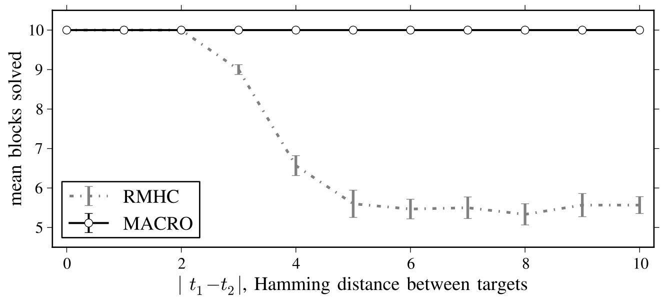

Fig. 5: Blocks solved in the SBB$_A$ problem as a function of distance between targets $h = |t_1 - t_2|$, for $k = 10, n = 100$. We tested MACRO-H and a random-mutation hill climber, allowing the former 4,000 evaluations and the latter 400,000 evaluations, and report data for the most successful mutation rate from $1/n$ to k/n. Data from 30 different SBB$_A$ instances for each value of $h \in [0, k]$. MACRO-H solves the problem in all instances, finding the solution to all blocks regardless of the distance between targets. Conversely, RMHC is unable to escape local optima except for small h.

the specific location of the targets. In contrast, hill-climbing can solve all blocks when the targets are close together, but as the distance increases the difficulty of moving between local optima causes performance to degrade substantially.

Considering these results together, we see that various model-building methods, are able to solve concatenated trap functions efficiently; and that MACRO-H is efficient on both problem classes. The blocks of the SBB* problem class are a discriminating factor; implying that the manner by which multi-scale search exploits discovered structural information is distinct from existing methods.

While TWOMAX and Trap are clearly closely related, the difficulty that they present comes from different factors. Specifically, the sub-problems in SBB are of fixed difficulty with increasing problem size; the challenge arises instead from the increasing distance ($k = \omega(\log n)$) between local optima that requires specific k-bit changes to escape. Conversely, when the sub-problems are small (*i. e.*, $k = O(\log n)$, *e. g.*, constant k), guessing the change required to escape a local optimum is feasible (or alternatively, to have the correct solution in an initial population). This latter case applies to the trap, which is difficult instead because the chance of finding the global optimum decreases with sub-problem size.

Two other logical extremes exist in this space of two-max-like problems. 1) Small sub-problems ($k = O(\log n)$, growing at most logarithmically with n, including constant k) that have a fixed probability of finding the global optimum. 2) Large sub-problems ($k = \omega(\log n)$, growing faster than logarithmically with n – such as $\sqrt{n}$) that have a diminishing probability of finding the global optimum with increasing problem size.

Scenario 1) is easier than a trap because there is no deception; any method that can solve a trap function will also be sufficient in this scenario. The sub-functions are small and so guessing the changes required to jump between local optima (in an unbiased/bitwise random manner) is sufficient.

Conversely, under scenario 2) the separation between local optima grows with problem size which rules out guessing the changes required; and yet, as the blocks get larger, the probability of finding the solution diminishes. By exhibiting more and more deception this parameterisation does not yield useful information to identify the global optimum, and yet guessing the answer through sampling would be inefficient.

In summary, the two features that make the trap and TWOMAX difficult sub-problems are distinct. MACRO-H is not efficient on a problem that exhibits both difficulties (scenario 2), although we do not know of a method that is capable of finding needles in (deceptive) haystacks efficiently. Problems with neither difficulty (scenario 1) can be solved efficiently by many algorithms, including MACRO-H. Section V proves that MACRO-H is efficient when one of the two difficulties is present – as for SBB or concatenated trap functions.

VII. HIERARCHICAL DIFFICULTY AND MULTI-SCALE SEARCH

The problems examined above have a transparent, one-layer structure that nonetheless presents substantial difficulties for various algorithmic approaches. When MACRO-H solves these problems it encapsulates the problem structure into composite units that are sufficient to solve the overall problem, when appropriately exploited. However, the two phases in each iteration of MACRO-H can operate on one another's outputs: the algorithm has the capacity to recurse. This ability is important because whereas the one-layer problems are separable, the hierarchical problems are not since higher-level interactions introduce dependencies between lower-level modules.

As noted, the ability to solve problems such as HXOR and hTrap in polynomial time has been shown both for multi-

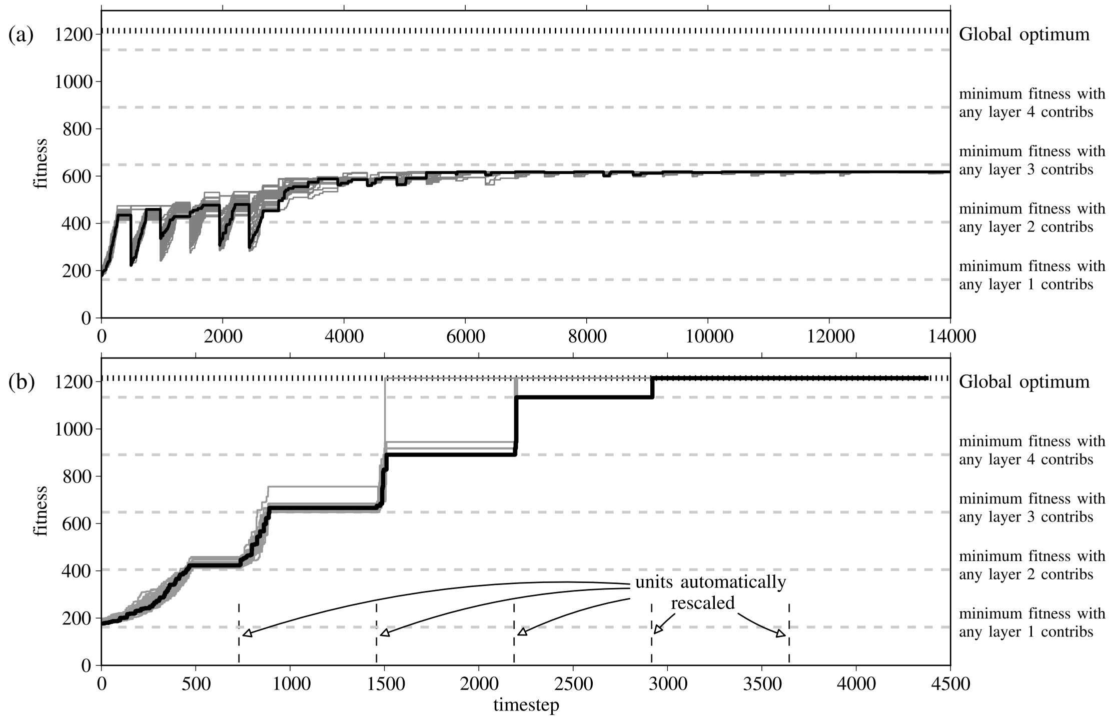

Fig. 6: Illustration of a 'two-scale' algorithm and a multi-scale search algorithm, on the H-SBB problem ($n = 243, \lambda = 5$). Conceptually, the operation of a single-scale algorithm is also illustrated by the first iteration of MACRO-H before the first rescaling (around timestep 750). We show the fitness over time, and each timestep corresponds to the advance of all members of the population (these are plotted simultaneously, and one arbitrary trajectory is highlighted). Data is from a single run that is representative of how each algorithm behaves. (a) a hybridised GA/hill-climber: After each round of crossover, a hill-climber is used to repair each solution (see [64]). This algorithm cannot exchange well-optimised building blocks from one individual to another and is therefore unable to search at higher levels of organisation, but the repeated restart of the hill-climber, focused on blocks that are not converged in the population, is useful in providing multiple chances at solving the bottom level. (b) MACRO-H; for the purposes of this figure, the final states of the UE-searchers from one round are retained as initial conditions in the subsequent round. In the first iteration of MACRO-H, the search phase uses primitive units, obtaining a distribution of different fitness values that correspond to various different local optima. These patterns are analysed to form composite units that correspond to the structure of the problem (see Algorithm 1). In subsequent iterations, the search phase uses these composite units and is able to escape what were local optima in the lower-scale problem (see Algorithm 2). Correctly exploiting these transformed units means that the search repeatedly rescales the space and reliably finds the global optimum.

scale search and for some EDAs. These problems do not discriminate between the abilities of these two approaches therefore. Like the results we showed for the concatenated trap problems, this is because these problems have small blocks.

Here, we define a hierarchical version of the SBB problem, the H-SBB, which brings together the hierarchical difficulty of HIFF (and derivatives) and the block scaling difficulty of SBB.

Definition 5. *Variables at one level are transformed before participating in the contribution function at the next level:*

$$\textsc{Transform}(x) = \begin{cases} 1 & \textit{if } x = 1^{|x|}, \\ 0 & \textit{if } x = 0^{|x|}, \\ \textit{NULL} & \textit{otherwise}, \end{cases}$$

The contribution function makes use of TWOMAX*, and only provides a non-zero value if all symbols are defined:*

$$\textsc{f}(x) = \begin{cases} 0 & \textit{if } \exists x[i] = \textit{NULL}, \\ \textsc{TwoMax}(x) & \textit{otherwise}, \end{cases}$$

The overall hierarchically consistent problem is controlled by λ, the number of layers, and n, and is defined as

$$\textsc{hsbb}(x) = \sum_{l=1}^{\lambda} k^{l-1} \sum_{i=1}^{n/k^l} \textsc{f}(\textsc{Transform}(x^{(ik^{l-1})})),$$

where $x^{(i)} = x[(i-1)k+1] \dots x[ik]$, and $x^{(ik)} = x[(i-1)k^2+1] \dots x[ik^2]$, and so on. We define the size of blocks at the bottom level, k, as $k = n^{1/\lambda}$. The fitness contribution of a block increases with the level to maintain the overall contribution of each level.

Note that this definition differs from HIFF [14], hTrap [34] and related derivatives, which use blocks comprising a fixed number of symbols, and a number of layers that scales as $\lambda = \log_k n$. In contrast, here we use blocks that scale as $\omega(\log n)$, and since λ and k are mutually constrained, a fixed number of layers λ allows $k = n^{1/\lambda}$ and is suitable for our purposes.

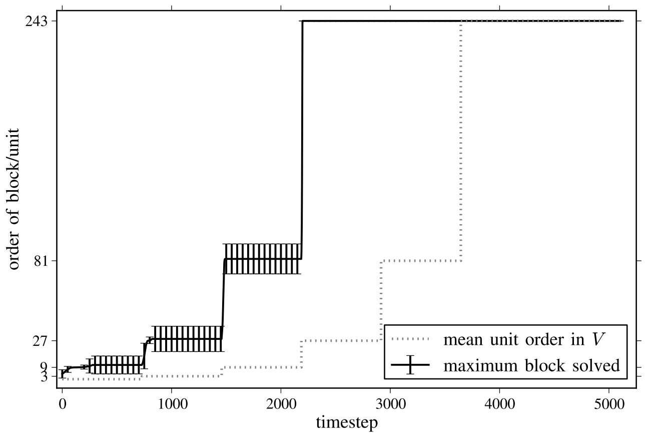

Fig. 7: The order (bits specified) of blocks discovered, and order of units in V, for MACRO on the H-SBB problem with $k=3, \lambda=5, n=243$. Block solutions are measured for every step of UE-SEARCH, and unit sizes are measured once per iteration. Data averaged over 30 repeats. We observe that MACRO solves the problem layer by layer: It uses blocks at one scale to discover solutions to blocks at the next scale. The co-occurrences among these solutions are used to form higher-order units, which then enables traversal of the next hierarchical scale. Note that the strict unit rescaling process results in homogeneous unit sizes within each iteration (and across all repeats). There is variance in the order of blocks, since occasionally a block more than one layer above is solved without using the gradient information at the current level.

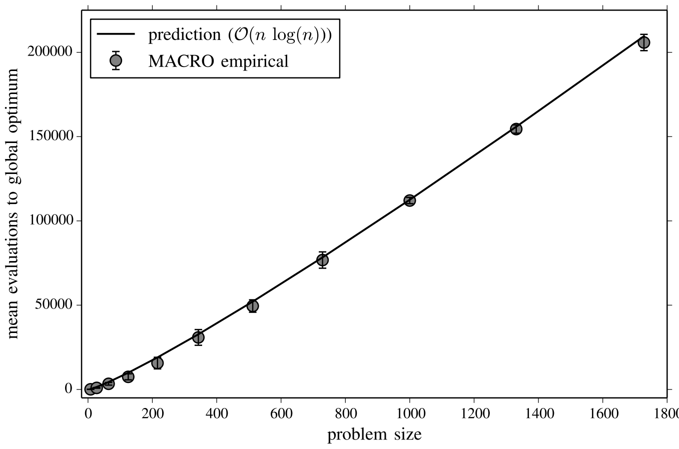

Fig. 8: MACRO finds the global optimum in the H-SBB problem with $k = n^{1/\lambda}, \lambda = 3$, in all 30 independent repeats. The solid line is a predicted runtime of $(\lambda - 1)td + t = O(n \log(n))$ function evaluations. The algorithm successively collapses the problem dimensionality into new units and exploits these new units to reliably obtain the global optimum in polynomial time.

A. Traversing hierarchy with repeated unit transformations

Here we briefly illustrate how MACRO-H behaves on the H-SBB problem. Figure 6 (b) shows the fitness progressing over time. The initial phase searches with the primitive units of the problem, which quickly becomes exhausted, finding several different optima. Once the units are rescaled (around timestep 750), search again rapidly improves in fitness. Each of the layers in this multi-layer problem is traversed successively through several recursive applications of the unit transformation (and subsequent exploitation). Figure 7 provides further insight into this dynamic, by examining the blocks that have been discovered, and how these become reflected in the unit set V. The units start out specifying a single bit, and as solutions to blocks at a given layer are discovered by UE-SEARCH, these are incorporated into larger units. These units are used in subsequent search, which enables discovery of module solutions at the next level, and so on.

Above we highlighted how the theoretical population sizing requirements for EDAs cause them to be inefficient on the SBB problem. Since the first layer of H-SBB tightly corresponds to SBB, we do not expect any algorithm that was inefficient on SBB to be efficient on this hierarchical variant. We do consider a particular kind of hybridised GA, which exemplifies what two-scale search can do that one-scale search does not achieve, thus providing a useful comparison for multi-scale search.

The algorithm uses uniform crossover, and applies a hill-climber to repair each of the children [64]. The crossover randomises the bits in blocks that disagree between the parents, and the disruption is repaired by the hill-climbing phase, effectively providing another chance to solve the block. Interestingly, this hybrid can solve the flat SBB* problem efficiently without any explicit model-building mechanisms, unlike EDAs (or multi-scale search). However, while it achieves 'two-scale search', Figure 6 (a) shows it is unable to recurse to higher levels as multi-scale search does. This algorithm does not create new units of variation that would enable it to increase the order of search over time, and the benefits would not be obtained when there are multiple layers to the problem.

On a related note, an EDA could also be hybridised, for instance by applying local search to the configurations drawn from the model, before selecting and subsequent model-building. This hybridisation may alleviate the theoretical requirements for population sizing, and thus more efficiently solve the first layer of the SBB problem. A univariate EDA such as UMDA can solve OneMax in polynomial time [65] (although a simple hill-climber compares favourably to univariate EDAs [66]). By analogy, we expect an EDA with a good model to be able to solve this module-max task. Superficially, such hybridisation may appear to approximate multi-scale search, but once again, without mechanisms to scale up the variational units or exploit those new units, once a model reflects the lowest level block structure, such an algorithm would not behave qualitatively differently from an unhybridised EDA.

B. Scalability

For the following experiment we use $\lambda = 3$ and $n \in [27, 1728]$ (with $k = n^{1/3} \in [3, 12]$). The H-SBB problem has the same number of blocks at the bottom level as an equally sized SBB instance, meaning that Theorem 1 provides a sufficient value for $d = O(\log n)$. This value is also sufficient for higher levels which have fewer blocks, and a constant setting is used throughout each experiment.

If we have units that represent block solutions of a given layer l, there are 2^k optima per $l+2$ block (and 2 per $l+1$

block). Learning units that represent block solutions to layer $l+1$, which are revealed by fitness gradients with layer l units, transforms the space such that only two optima remain. Since the problem is hierarchically consistent, each time the units are transformed, the space appears to be the same but in lower dimensionality. Thus, if the algorithm is able to transform the space from one layer to the next in polynomial time, it should be able to solve each of the layers in turn to find a global optimum in time that is polynomial overall. In Section V we proved that MACRO-H can transform one layer to solve the SBB problem. Provided that this analysis transfers to this multi-level problem, we should expect the overall cost to scale as $O(\lambda t(n)\log(n)) = O(n\log(n))$ since $t(n) = O(n)$ and λ is constant. The data shown in Figure 8 confirms this trend.

Hierarchical landscapes emphasise the capability of MACRO-H to recursively transform the space into lower-dimensional problems through multiple scales of organisation.

VIII. Discussion

A. *Extensions of* MACRO*, and other problem classes*

While Corollary 1 guarantees excellent performance on a wide range of problems it also defines clear limits. All subfunctions of the $(n/k, k)$-separable objective functions can have at most two local optima. Problems consisting of sub-problems with more than two optima are not necessarily solvable by MACRO-H, but Section VII demonstrates that MACRO-H can solve these also in the particular case where large multi-peaked sub-problems can be decomposed hierarchically.

In general the limitations of MACRO-H include cases where false dependencies are inferred and cases where true dependencies are missed. The latter can be alleviated by the soft joins version, MACRO-S, where a join is formed with a probability that is proportional to the number of times search units have been found jointly in the underlying UE-Search. MACRO-S is able to recover the structure and form probabilistic associations even with lower-quality information [16]. While this work speaks to the robustness of the general approach to dependency identification, it does not tightly define the scope of problems for which multi-scale search is well suited. It is clear that our investigation using a single family of problems cannot support arguments for or against generality (although see Section II, which describes a broader set of multi-scale search results). This area deserves significant further effort, but is beyond the scope of the present paper.

Multi-scale search emphasises learning structure from local optima (at each scale). Accordingly, to successfully recover structural information, the local optima must contain information about the global optimum. It is clear that this is the case in the SBB, since the modules are separable. However, the extent to which sub-patterns that are common across many local optima are also common to the global optimum is likely to be pivotal in determining the applicability of a multi-scale search approach. Our work elsewhere has investigated this question more formally [17].

The most basic unit in MACRO, the primitive, specifies a single value for a given variable. The two primitives corresponding to a particular variable have no special properties that represent an explicit awareness that they directly compete for a specific variable. This representation enables MACRO to discard unnecessary values from the set of units it learns. Such representation may be important for domains in which the alternative primitives do not obviously compete directly for specific variables (e. g., subfunctions in genetic programming [43] or edges in permutations spaces like the travelling salesman problem). In some such problem domains, partial evaluations may be feasible. This may yield valuable information about credit assignment, enabling greater efficiency in identifying interdependency structures. Note, however, that MACRO-H successfully learns problem structures without access to partial evaluations. However, representing variables as the basic unit may be more appropriate for the binary combinatorial optimisation domain. Elsewhere, we have used variables as the basic unit [17].

There are many variant algorithms incorporating the general principles of multi-scale search yet to explore. For instance, we could replace UE-SEARCH with a more sophisticated protocol, provided it is able to exploit composite variation. Alternatives include: a GA; simulated annealing; or even an instance of MACRO itself as each of the unit-exploiting searchers. Our results have shown that because other model-building do not use learned problem structure to define new variational units, they are not exploiting learned structure in the same manner as multi-scale search. Interestingly, our approach of higher-order search is entirely compatible with the advanced model-building techniques used by state-of-the-art EDAs. Where we have used elementary dependency detection, there is scope for incorporating something more akin to a Bayesian network as used in BOA [34] and EBNA [67], or to employ a probing method, which can efficiently discover the linkage structure even when building blocks overlap (provided the interactions are of low order) [68]. Likewise, these or other EDAs could use their model to inform trajectories of higher-order variation steps, before any further model-building is applied.

B. *Hierarchical transitions in biological evolution*

The mechanisms detailed in this paper may appear to be quite far removed from biological processes, but (as with all good ideas, it seems) biological evolution did it first. MACRO employs global procedures and data structures to identify and utilise problem structure. However, the essence of multi-scale search is really very simple: entities that were accidentally correlated (due to fitness dependencies in the problem) become canalised or deliberately correlated (due to learned associations). This is actually what one would expect to happen if associations were optimised to maximise robustness (see also [**?**], [26]). We have been investigating fully-distributed individual-based versions of these processes, *i. e.*, an ecosystem in which natural selection evolves symbiotic associations between species, and we find that the associations that evolve match those learned by MACRO [49], [69]. This means that when natural selection can act on individual traits that affect social relationships [70] to, in effect, create new units of selection, they will implement multi-scale search in a bottom-up emergent manner [17], [69]. The hard joins version of MACRO is analogous to the formation of explicit higher-level selective units (as per the major evolutionary transitions [71]), but the soft-joins version of MACRO shows that reforming specific (symbiotic) groups on-the-fly is sufficient and algorithmically equivalent. Thus, in identifying what multi-scale search can solve that single-scale search cannot in this paper, we also show that such biological processes are fundamentally a different class of algorithm from micro-scale evolutionary processes.

IX. Conclusions

Multi-scale search is an evolutionary process based on variation and selection where the units of variation are re-scaled by canalising combinations of existing variational units. In this paper we have analysed the runtime performance of a multi-scale search algorithm, MACRO-H, on an exactly separable class of problem, the SBB, which has not previously been shown to be solvable in polynomial time. With this work we are the first to show that multi-scale search has an algorithmic niche that is not occupied by other approaches. Moreover, by capturing the modular structure of a separable problem explicitly, MACRO-H, is able to recurse through successive levels of organisation. Accordingly, MACRO-H, is also apt at solving hierarchically modular problems, even though these problems are not separable and exhibit fitness dependencies at many scales up to and including n bits.

The basic intuition of the BBH, that good combinations of bits can be found and exploited to identify modules, and then good combinations of modules can be found and exploited to identify solutions to the next level of problem structure, has been difficult to demonstrate. EDAs have been successful in identifying problem structure in random-linkage building block problems and using this information to bias the distribution of samples explored. But multi-scale search uses discovered structure in a different way – to explicitly search combinations of discovered modules, and can apply this approach through multiple levels of hierarchical organisation. This approach demonstrates that the intuition behind the BBH, developed for GAs with tight-linkage, can be exploited successfully even in problems with random linkage structure.

This paper has focused on proving that multi-scale search is qualitatively and quantitatively different from existing approaches because of the way in which a simple search process is explicitly re-instantiated at successively higher levels of organisation. This provides a formal basis for extensions that can be applied to other problem classes and for investigating their implications for hierarchical processes in biological evolution.

References

[1] W. T. McCormick, Jr., P. J. Schweitzer, and T. W. White, "Problem decomposition and data reorganization by a clustering technique," *Oper Res*, vol. 20, no. 5, pp. 993–1009, 1982.

[2] D. Smith, "Top-down synthesis of divide-and-conquer algorithms," *Artif Intell*, vol. 27, no. 1, pp. 43–96, 1985.

[3] C. Y. Baldwin and K. B. Clark, *Design rules, Vol 1: The power of modularity*. MIT Press, 2000.

[4] H. A. Simon, *The Sciences of the Artificial*. MIT Press, 1969.

[5] D. Michie, "Memo functions and machine learning," *Nature*, vol. 218, pp. 19–22, 1968.

[6] J. E. Laird, P. S. Rosenbloom, and A. Newell, "Chunking in soar: The anatomy of a general learning mechanism," *Mach Learn*, vol. 1, no. 1, pp. 11–46, 1986.

[7] R. Bellman, *Dynamic programming*. Princeton University Press, 1957.

[8] J. A. Walker and J. F. Miller, "The automatic acquisition, evolution and reuse of modules in cartesian genetic programming," *IEEE T Evolut Comput*, vol. 12, no. 4, pp. 397–417, 2008.

[9] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol Comput*, vol. 8, no. 1, pp. 1–29, 2000.

[10] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE T Evolut Comput*, vol. 11, no. 6, pp. 712–731, 2007.

[11] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[12] D. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[13] M. Mitchell, S. Forrest, and J. H. Holland, "The royal road for genetic algorithms: Fitness landscapes and GA performance," in *Procs. ECAL*, pp. 245–254, 1992.

[14] R. A. Watson, G. S. Hornby, and J. B. Pollack, "Modelling building-block interdependency," in *PPSN V*, pp. 97–106, 1998.

[15] R. A. Watson and T. Jansen, "A building-block royal road where crossover is provably essential," in *Procs. GECCO*, pp. 1452–1459, 2007.

[16] R. Mills, *How Micro-Evolution Can Guide Macro-Evolution: Multi-Scale Search via Evolved Modular Variation*. PhD thesis, University of Southampton, 2010.

[17] R. A. Watson, R. Mills, and C. Buckley, "Transformations in the scale of behaviour and the global optimisation of constraints in adaptive networks," *Adap Beh*, vol. 19, no. 4, pp. 227–249, 2011.

[18] D. Iclanzan and D. Dumitrescu, "Overcoming hierarchical difficulty by hill-climbing the building block structure," in *Procs. GECCO*, pp. 1256–1263, 2007.

[19] J. Houdayer and O. C. Martin, "A hierarchical approach for computing spin glass ground states," *Phys. Rev. E*, vol. 64, no. 5, p. 056704, 2001.

[20] K. Mahdavi, M. Harman, and R. M. Hierons, "A multiple hill climbing approach to software module clustering," in *Procs. IEEE Conference on Software Maintenance*, pp. 315–324, 2003.

[21] D. Thierens, "The linkage tree genetic algorithm," in *Procs. PPSN XI*, pp. 264–273, 2010.

[22] T. Gross and B. Blasius, "Adaptive coevolutionary networks: a review," *J R Soc Interface*, vol. 5, no. 20, pp. 259–271, 2008.

[23] D. Iclanzan and D. Dumitrescu, "Towards memoryless model building," in *Procs. GECCO*, pp. 2147–2152, 2008.

[24] R. Mills, R. A. Watson, and C. Buckley, "Emergent associative memory as a local organising principle for global adaptation in adaptive networks," in *Procs. ICCS*, pp. 417–430, 2011.

[25] R. A. Watson, C. Buckley, and R. Mills, "Optimisation in 'self-modelling' complex adaptive systems," *Complexity*, vol. 16, no. 5, pp. 17–26, 2010.

[26] R. A. Watson, R. Mills, and C. Buckley, "Global adaptation in networks of selfish components: emergent associative memory at the system scale," *Artif Life*, vol. 17, no. 3, pp. 147–166, 2011.

[27] R. A. Watson, *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, Brandeis University, Waltham, MA, 2002.

[28] K. Deb and D. E. Goldberg, "Analyzing deception in trap functions," in *Foundations of Genetic Algorithms 2* (L. D. Whitley, ed.), pp. 93–108, Morgan Kaufmann, 1993.

[29] D. Goldberg, K. Sastry, and T. Latoza, "On the supply of building blocks," in *Procs. GECCO*, pp. 336–342, 2001.

[30] M. Pelikan, *Bayesian optimisation algorithm: from single level to hierarchy*. PhD thesis, University of Illinois at Urbana-Champaign, 2002.

[31] D. Whitley, K. Mathias, S. Rana, and J. Dzubera, "Building better test functions," in *Procs. ICGA*, pp. 239–246, 1995.

[32] T. Jones, "Crossover, macromutation, and population-based search," in *Procs. ICGA*, pp. 73–80, 1995.

[33] P. Larrañaga and J. A. Lozano, eds., *Estimation of distribution algorithms. A new tool for evolutionary computation*. Boston: Kluwer, 2002.

[34] M. Pelikan and D. E. Goldberg, "Hierarchical problem solving and the Bayesian optimization algorithm," in *Procs. GECCO*, pp. 267–274, 2000.

[35] U. Aickelin, E. K. Burke, and J. Li, "An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering," *J Oper Res Soc*, vol. 58, pp. 1574–1585, 2007.

[36] M. Pelikan and D. E. Goldberg, "Hierarchical BOA solves Ising spin glasses and MAXSAT," in *Procs. GECCO*, pp. 1271–1282, 2003.

[37] R. Santana, P. Larrañaga, and J. Lozano, "Protein folding in simplified models with estimation of distribution algorithms," *IEEE T Evolut Comput*, vol. 12, pp. 418–438, 2008.

[38] R. A. Watson, D. Weinrich, and J. Wakerley, "Genome structure and the benefit of sex," *Evolution*, vol. 65, pp. 523–536, 2011.

[39] R. Impagliazzo, R. Paturi, and F. Zane, "Which problems have strongly exponential complexity?," *J. Comput Syst Sci*, vol. 63, pp. 512–530, 2001.

[40] T.-L. Yu and D. E. Goldberg, "Conquering hierarchical difficulty by explicit chunking: Substructural chromosome compression.," in *Procs. GECCO*, pp. 1385–1392, 2006.

[41] J. P. Rosca, *Hierarchical Learning with Procedural Abstraction Mechanisms*. PhD thesis, University of Rochester, 1997.

[42] P. Angeline and J. B. Pollack, "Coevolving high-level representations," in *Artificial Life III* (C. G. Langton, ed.), pp. 55–71, Addison-Wesley, 1994.

[43] J. R. Koza, *Genetic programming: On the programming of computers by means of natural selection*. MIT Press, 1992.

[44] R. A. Watson and J. B. Pollack, "A computational model of symbiotic composition in evolutionary transitions," *Biosystems*, vol. 69, no. 2–3, pp. 187–209, 2003.

[45] E. D. de Jong, R. A. Watson, and D. Thierens, "On the complexity of hierarchical problem solving," in *Procs. GECCO*, pp. 1201–1208, 2005.

[46] A. Defaweux, *Evolutionary transitions as a metaphor for compositional search*. PhD thesis, Vrije Universiteit Brussel, 2006.

[47] R. Mills and R. A. Watson, "Symbiosis, synergy and modularity: Introducing the reciprocal synergy symbiosis algorithm," in *Procs. ECAL*, pp. 1192–1201, 2007.

[48] R. Mills and R. A. Watson, "Variable discrimination of crossover versus mutation using parameterized modular structure," in *Procs. GECCO*, pp. 1312–1319, 2007.

[49] R. A. Watson, N. Palmius, R. Mills, S. Powers, and A. Penn, "Can selfish symbioses effect higher-level selection?," in *Procs. ECAL*, pp. 27–36, 2009.

[50] T. Jansen and R. P. Wiegand, "The cooperative coevolutionary (1+1) EA," *Evol Comput*, vol. 12, no. 4, pp. 405–434, 2004.

[51] M. Pelikan and D. E. Goldberg, "Genetic algorithms, clustering, and the breaking of symmetry," in *Procs PPSN VI*, pp. 385–394, 2000.

[52] T. Friedrich, P. S. Oliveto, D. Sudholt, and C. Witt, "Analysis of diversity-perserving mechanisms for global optimisation," *Evol Comput*, vol. 17, no. 4, pp. 455–476, 2009.

[53] D. E. Goldberg, C. Van Hoyweghen, and B. Naudts, "From TwoMax to the Ising model: Easy and hard symmetrical problems," in *Procs. GECCO*, pp. 626–633, 2002.

[54] S. Droste, T. Jansen, and I. Wegener, "Upper and lower bounds for randomized search heuristics in black-box optimization," *Theory of Computing Systems*, vol. 39, no. 4, pp. 525–544, 2006.

[55] S. Droste, T. Jansen, K. Tinnefeld, and I. Wegener, "A new framework for the valuation of algorithms for black-box optimization," in *Foundations of Genetic Algorithms 7 (FOGA)*, pp. 253–270, Morgan Kaufmann, 2003.

[56] P. K. Lehre and C. Witt, "Black-box search by unbiased variation," *Algorithmica*, vol. 64, pp. 623–642, 2012.

[57] I. Wegener, "Simulated annealing beats metropolis in combinatorial optimization," in *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 589–601, 2005.

[58] S. Nijssen and T. Back, "An analysis of the behavior of simplified evolutionary algorithms on trap functions," *IEEE T Evol Comput*, vol. 7, no. 1, pp. 11–22, 2003.

[59] G. R. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," *Evol Comput*, vol. 7, no. 3, pp. 231–253, 1999.

[60] T.-L. Yu, K. Sastry, D. E. Goldberg, and M. Pelikan, "Population sizing for entropy-based model building in estimation of distribution algorithms," in *Procs. GECCO*, pp. 601–608, 2007.

[61] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The bayesian optimization algorithm," in *Procs. GECCO*, pp. 525–532, 1999.

[62] M. Pelikan, K. Sastry, and D. E. Goldberg, "Sporadic model building for efficiency enhancement of the hierarchical BOA," *Genetic Programming and Evolvable Machines*, vol. 9, pp. 53–84, 2008.

[63] S. Shakya, R. Santana, and J. A. Lozano, "A markovianity based optimisation algorithm," *Genetic Programming and Evolvable Machines*, vol. 13, no. 2, pp. 159–195, 2012.

[64] A. Prügel-Bennett, "Benefits of a population: five mechanisms that advantage population-based algorithms," *IEEE T Evolut Comput*, vol. 14, no. 4, pp. 500–517, 2010.

[65] C. González, *Contributions on Theoretical Aspects of Estimation of Distribution Algorithms*. PhD thesis, University of the Basque Country, San Sebastián, Spain, 2005.

[66] T. Chen, P. K. Lehre, T. Ke, and X. Yao, "When is an estimation of distribution algorithm better than an evolutionary algorithm?," in *Procs. IEEE CEC*, pp. 1470–1477, 2009.

[67] R. Etxeberria and P. Larrañaga, "Global optimization using Bayesian networks," in *Procs. 2nd Symposium on Artificial Intelligence (CIMAF 99)*, pp. 332–339, 1999.

[68] R. Heckendorn and A. H. Wright, "Efficient linkage discovery by limited probing," *Evol Comput*, pp. 517–545, 2004.

[69] R. A. Watson, A. Jackson, N. Palmius, R. Mills, and S. T. Powers, "The evolution of symbiotic partnerships and their adaptive consequences," submitted.

[70] S. T. Powers, A. S. Penn, and R. A. Watson, "The concurrent evolution of cooperation and the population structures that support it," *Evolution*, pp. 1527–1543, 2011.

[71] J. Maynard Smith and E. Szathmáry, *The Major Transitions in Evolution*. Oxford University Press, 1995.

PLACE PHOTO HERE

Rob Mills received the M. Eng (2005) and Ph. D. (2010) degrees from University of Southampton, UK, where he is presently Postdoctoral Fellow. He has previously held research posts at University of Oxford and ARM ltd. He has published 6 journal articles and 12 conference papers, and holds two patents. His research interests, bridging evolutionary computing and theoretical biology, focus on the organisation of interactions in evolving systems, and information processing in natural systems.

PLACE PHOTO HERE

Thomas Jansen received the Diploma and Ph. D. degrees in Computer Science from the Technical University Dortmund, Germany, in 1996 and 2000, respectively. His Ph. D. thesis on the theoretical analysis of evolutionary algorithms was awarded the University Best Dissertation Award.

He was Post-doc researcher at the George Mason University 2001–2002, Juniorprofessor for Computational Intelligence at the Technical University of Dortmund 2002–2009, Stokes Lecturer at the University College Cork, Ireland, 2009–2012 and is Senior Lecturer at Aberystwyth University, UK, since 2013. He has authored 20 journal papers, 41 conference articles, 6 book chapters and one text book on Analyzing Evolutionary Algorithms. He is Senior Member of the ACM and associate editor of *Evolutionary Computation* and *Artificial Intelligence*.

PLACE PHOTO HERE

Richard A. Watson was awarded his Ph.D. in Computer Science from Brandeis University (USA) in 2002. His background in computational models of evolution was complemented by a fellowship at Harvard University in the Dept. of Organismic and Evolutionary Biology. Since 2004 he has been Senior Lecturer at University of Southampton (UK). In 2006 he received the IEEE international award "Ten to Watch in Artificial Intelligence". His publications span topics including artificial life, robotics, evolutionary modelling, evolutionary computation, and computational biology; and he is author of "Compositional Evolution: The Impact of Sex, Symbiosis, and Modularity on the Gradualist Framework of Evolution" with MIT Press. His current research focuses on understanding the algorithmic principles of natural processes that create adaptive biocomplexity.