Applications of bioinspired computing (to real world problems) manuscript No.
(will be inserted by the editor)

Deep Optimisation: Multi-Scale Evolution by Inducing and Searching in Deep Representations

***Jamie Caldwell · Joshua Knowles · Christoph Thies · Filip Kubacki · Richard Watson**

Received: date / Accepted: date

Abstract We investigate the optimisation capabilities of an algorithm inspired by the Evolutionary Transitions in Individuality. In these transitions, the natural evolutionary process is repeatedly rescaled through successive levels of biological organisation. Each transition creates new higher-level evolutionary units that combine multiple units from the level below. We call the algorithm Deep Optimisation (DO) to recognise both its use of deep learning methods and the multi-level rescaling of biological evolutionary processes. The evolutionary model used in DO is a simple hill-climber, but, as higher-level representations are learned, the hill-climbing process is repeatedly rescaled to operate in successively higher-level representations. The transition process is based on a deep learning neural network (NN), specifically a deep auto-encoder. Our experiments with DO start with a study using the NP-hard problem, multiple knapsack (MKP). Comparing with state-of-the-art model-building optimisation algorithms (MBOAs), we show that DO finds better solutions to MKP

Jamie Caldwell (*Corresponding Author)
Agents, Interaction and Complexity (AIC) Research Group,
Southampton University, Southampton, SO17 1BJ, UK
E-mail: j.r.caldwell@soton.ac.uk
Telephone Number: (+44) 02380 592690
orcidID: 0000-0002-6531-2289

Christoph Thies
Agents, Interaction and Complexity (AIC) Research Group,
Southampton University, Southampton, SO17 1BJ, UK

Filip Kubacki
Agents, Interaction and Complexity (AIC) Research Group,
Southampton University, Southampton, SO17 1BJ, UK

Richard Watson
Agents, Interaction and Complexity (AIC) Research Group,
Southampton University, Southampton, SO17 1BJ, UK

Joshua Knowles
School of Computer Science, University of Birmingham, B15 2TT, UK
Present address: Invenia Labs, Cambridge, CB2 1AW, UK

instances and does so without using a problem-specific repair operator. A second, much more in-depth investigation uses a class of configurable problems to understand more precisely the distinct problem characteristics that DO can solve that other MBOAs cannot. Specifically, we observe a polynomial vs exponential scaling distinction where DO is the only algorithm to show polynomial scaling for all problems. We also demonstrate that some problem characteristics need a deep network in DO. In sum, our findings suggest that the use of deep learning principles have significant untapped potential in combinatorial optimisation. Moreover, we argue that natural evolution could be implementing something like DO, and the evolutionary transitions in individuality are the observable result.

Keywords Model-Building Optimisation Algorithms · Deep Autoencoder · Multi-Scale Search · Problem Structure

1 Declarations

Funding: We acknowledge financial support from the EPSRC Centre for Doctoral Training in Next Generation Computational Modelling grant EP/L015382/1.
Conflicts of interest/Competing interests: The authors declare that they have no conflict of interest.
Availability of data and material: Not Applicable
Code availability: Avaliable on request.
Authors' contributions:
Ethics approval: Not Applicable
Consent to participate: Not Applicable
Consent for publication: Not Applicable

2 Introduction

Deep Optimisation (DO) is a recent addition to the class of Model-Building Optimisation Algorithms (MBOA) that exploits deep learning concepts [6, 5]. MBOAs are a class of black-box optimisation techniques inspired by the process of variation and selection in natural evolution. MBOAs work by adapting the solution neighbourhood using a machine learning model to capture relationships within a distribution of promising solutions. These relationships form a compressed representation of the search space allowing variation to explore in a redefined neighbourhood and find higher quality solutions. Exploiting problem structure in this way has been successful in multiple problem domains [33, 1, 7, 41]

There exist multiple MBOAs that show state-of-the-art performance [36, 46, 16, 22]. However, as we discuss in this paper, the model used to adapt the solution neighbourhood and the method used to then explore this neighbourhood differ between the algorithms. However, to the best of our knowledge, there do not exist known characteristics of problems, such that one (existing state-of-the-art MBOA) algorithm can handle efficiently while another cannot. We also observe that the presently existing state-of-the-art algorithms do not use a neural network model. MBOAs that

do use a neural network as the model have failed to show state-of-the-art performance [10, 38, 40]. Against this (briefly described) background, this paper aims to answer: Can a deep neural network model improve the performance of a MBOA? Further, we aim to understand what types of problem structure separate performance between MBOAs (including our new one) into can and cannot do.

In this paper, we introduce[1] the Deep Optimisation algorithm (DO), which uses a deep autoencoder model to encode relationships between problem variables. DO differs from the existing MBOAs in two important ways. Firstly, DO constructs a deep representation of the solution by recursively transforming the solution representation. This is performed using layerwise learning, where each layer is constructed by recognising correlations in a distribution of solutions that are locally optimal relative to the neighbourhood defined by the preceding layer. Each layer induced by the autoencoder transforms the representation of the solution, presenting a solution space with a higher-level of organisation. Specifically, a change to a single variable in the transformed representation corresponds to an organised change to multiple variables to the solution representation. Secondly, previous MBOAs that use a neural network model generate complete solutions from the model — Model-Informed Generation (MIG). In contrast, DO improves a solution through an iterative search process with steps that are informed by the model, i.e. through small changes in the latent representation — Model-Informed Variation (MIV).

DO, like other MBOAs, is inspired by the processes of biological evolution. However, whereas other methods aim to model the structure of allelic associations in a population of individuals at a single level of biological organisation, DO is inspired by evolutionary processes operating over multiple levels of organisation. Such Evolutionary Transitions in Individuality (ETIs) [56], have the characteristic that multiple individuals at one level of organisation form associations that result in a new evolutionary unit at a higher level of organisation [42], e.g. the transition from unicellular life to multicellular organisms. These are not merely cooperative relationships among coevolving unicellular organisms. They are new evolutionary units that allow multiple units from the previous level of organisation to be combined and selected together, enabling evolution to move through solution space at a new level of representation [31, 55]. In DO, a deep network learns to encode solutions into a compressed latent space. Small variations in this latent space are decoded back to the solution space, producing large organised variations to the original problem variables. By using MIV, in an iterative selection process (local search in the latent space), DO represents the process of variation and selection acting on high-order evolutionary units. Each subsequent layer recodes again, in the analogue of successive hierarchical transitions in individuality [51].

Alternative methods that use machine learning (ML) to improve optimisation include: learning a heuristic for a set of problem instances [58, 24, 2]; learning to combine a set of given heuristics (hyperheuristics) [45]; using a surrogate model to approximate the fitness function [49], including 'Bayesian optimization' in the ML community [43, 19]; adapting the learning function to bias future search [20, 4];

[1] We follow the convention that this journal article is the introducing/defining work, notwithstanding that a preprint version and short conference paper have described the algorithm in outline previously — see references.

embedding a machine learning model within the model of a combinatorial problem [25] and using machine learning to select a suitable solver [48]. In continuous problem spaces, learning to adapt a variation operator from a population samples has been an important advance [17] and is increasingly based on manifold learning (information geometry) approaches [32, 57]. Back in combinatorial spaces, the cross-entropy method [12] also learns to converge its samples based on information geometric principles, and, relatedly, ant colony optimisation algorithms solve combinatorial problems by a constructive version of a simple MBOA [59]. The use of deep reinforcement learning (RL) algorithms for combinatorial optimisation is a popular approach [29]. Deep reinforcement learning is used to learn a policy that performs an action on a given state to improve the solution. This policy can then be used on multiple instances from the same problem class. Unlike DO and other MBOAs, these methods do not use the model to recode the neighbourhood of a search space into a higher-level representation.

In this paper, we investigate what type of problem characteristics DO can overcome that other MBOAs cannot. We first evaluate the performance of MBOAs and DO on the multi-dimensional Knapsack problem, and show that DO can find solutions that other MBOAs cannot. We then explore types of problem characteristics, by using a configurable problem with controllable structure to distinguish the capabilities of MBOAs and DO. We show that their exists problem structure separates the performance of the different models used, different methods for exploring the reorganised neighbourhood and ability to induce a deep representation.

3 The Deep Optimisation Algorithm

In this section, we first provide a high-level motivation for the architecture of DO and then give the specific details of how the algorithm works. Detailed comparisons of DO's architecture and procedures with related methods (notably other MBOAs) are left to Section 4.

Intuitively, we might learn to use a language by first recognising the combinations of sounds that make up common syllables, then combinations of syllables that make up common words, and so on through phrases and sentences. With each rescaling of the representation, we are able to construct meaningful utterances at a higher level of organisation more easily. This, in turn, provides a better signal for us to learn the next level of structure. Only with learning of skills at all of these levels is it possible for an author to learn how to construct an effective essay or build a narrative arc; search in the space of individual letters would be useless even if guided by an accurate objective function. Deep Optimisation uses a similar intuition to learn multiple levels of representation, making it increasingly easy to construct high quality solutions to an optimisation problem, as it learns how to combine existing units together in useful ways. An evolutionary unit is an entity subjected to variation and selection and although its fitness may be sensitive to the other units it is evaluated with, its variation and reproduction is, at least initially, independent from other evolutionary units. In terms of optimisation, an evolutionary unit represents any unit of variation — a partial solution [50] — that can be selected and combined with others to form a

whole solution. For example, a change made to a single solution variable is the lowest level evolutionary unit, and a change made to multiple solution variables simultaneously (which is retained or discarded together) is a higher-level unit. At the start of the DO algorithm, each solution variable is a separate evolutionary unit. A transition describes a transformation process that induces higher-order evolutionary units. The transformation performs a coordinate restructuring of a single-unit change made to a solution, enabling a single-unit change to perform a simultaneous change to multiple solution variables before selection acts. Consequently, evolution is rescaled to operate at a higher level of organisation that facilitates informed movements between solutions that were far apart in the original solution space [54].

This idea of rescaling the search process from searching combinations of bits to searching combinations of modules at multiple levels (higher-levels) of organisation is familiar in many areas of engineering and in genetic algorithms [31, 13]. However, building-block ideas in genetic algorithms were inspired by the biology of crossover in sexual reproduction which suffers from the need to represent the linkage of variables in a linear chromosome (a limitation largely responsible for the motivation of MBOAs and their departure from the biology that inspired the genetic algorithm). DO takes a different inspiration — specifically, from the ETIs and not crossover. Each new level of representation in DO represents the formation of new evolutionary units that in turn becomes subject to evolutionary processes. These new units can then form relationships to create further levels of organisation as the next level is learned, and so on, in the analogue of successive hierarchical transitions in individuality [51]. Thus a solution is now described by a set of high-level features rather than the original problem variables.

The term transition is used in DO to describe the process of inducing higher-order evolutionary units from the existing set of lower-level evolutionary units. The components that make up the transition processes are:

1. Discovering how existing (lower-level) units combine — providing a signal for learning a compressed representation.
2. Inducing a compressed representation of these combinations - creating new units at a higher-level of organisation.
3. Rescaling the process of variation and selection to operate using the higher-order units — to search in the space defined by the compressed representation.

The following subsections describe these three components. For clarity, note that the optimisation process (evolutionary search), is separated from the learning process; the evolutionary process is not used to evolve a neural network as in NeuroEvolution [44]. In DO, the compressed representation is induced (by standard ML methods) from information discovered by the evolutionary search using the lower-level organisation, and the results are used to instantiate a new evolutionary process at the higher level of representation. Also note that, although a case can be made that biological ETIs naturally implement a form of DO (as is our inspiration [55, 51]), here our implementation uses abstract ML methods that do not intend to model that biology in a direct way.

3.1 Discovering how lower-level units combine:

In familiar learning tasks, such as classification and regression tasks, a model is learned from a data set. In an optimisation problem, no such data set is provided. We aim to learn information about the problem structure (which problem variables 'go together') but this information is only implicit in the fitness function (or objective function of the problem). To create a data set from which this information can be discovered, we must first gain some visibility of it by 'probing' the problem. In DO, this is provided by using a local search process (a simple evolutionary process) where a fitness function guides the search process to locally optimal solutions. Although this process is uninformed (knows nothing about the problem structure except that provided by selection) a distribution of solutions from different starting positions can be used as a training set to extract useful information about the problem structure implicit in the fitness function.

Hill-climbing (also known as a "(1+1) evolutionary algorithm") is a simple evolutionary process sufficient for this. Fig. 1 illustrates a fitness landscape of an optimisation problem. For most optimisation problems, the fitness landscape is unknown and is used here only to help demonstrate the idea. The x and y axes represent the solution space and the z-axis represents the fitness of each solution. Hill-climbing is an efficient method to exploit local information about the search space but is susceptible to becoming trapped at sub-optimal solutions (a locally optimal solution). In order to escape a local optimum, a solution can be reset in a new but random region of the solution space. However, the solution is likely to become trapped at a different local optimum. In complex problems, the likelihood of finding a globally optimal solution via resets becomes exponentially small as the number of problem dimensions increases.

Instead, an alternative approach to escaping a local optimum is to make a large change to the solution, for example a multi-variable substitution in the case of a binary problem. Allowing multi-variable substitutions effectively transforms the neighbourhood of a solution space such that solutions that were initially far apart become adjacent. A multi-variable substitution could be performed by making a random change to a sub-set of the solution variables. However, the likelihood of this producing an adaptive change vanishes as the size of the multi-variable substitution increases. Instead, one requires a multi-variable substitution that is *informed*, i.e. has knowledge of the problem structure, and is consequently more likely to improve the fitness of a solution by enabling a large but coordinated change. The type of multi-variable substitution that will be adaptive is, however, dependent on the problem instance — and it is this information that DO will learn.

A distribution of locally-optimal solutions, found by searching combinations of units at the current level of organisation, contains combinations of variables that work well together to some degree. The dimensionality of the variation observed in this distribution is less than the original dimensionality of possible variation. Consequently, a dimensional compression of the distribution of solutions can be performed to induce higher-order units of variation — extracting features of variation that appear in the distribution of locally-optimal solutions. This much is a fairly standard approach in MBOAs (although we use the technique of learning from a distribution of locally op-

timal solutions [23] rather than the more common practice of using a selected subset from a population of randomly generated solutions). But importantly, DO represents this compression using a standard auto-encoder, enabling well-developed ML techniques to be applied and permitting multiple levels of deep structure to the learned.

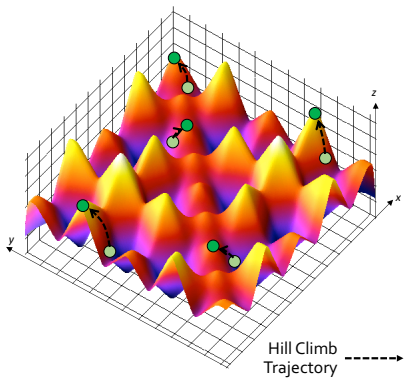


Fig. 1 Hill-climbing in a search space can efficiently exploit local information about the problem structure, but can easily become trapped at a sub-optimal solution (a local optimum). The search process can be repeatedly reset to random start points (light circles) to explore different regions of the search space and provide a diverse distribution of locally optimal solutions (dark circles)

3.2 Inducing a compressed representation

Learning a compressed representation of a distribution of solutions found at the current level of organisation permits a transformation of the space of solutions. The dimensional reduction will represent the data-set by extracting features that represent the salient units of variation in the data-set. For instance, a regular relationship between variables will be extracted and represented by a variable in the higher level representation. Consequently, differences between solutions are described as differences between features. Therefore, initially, far apart solutions (differing in the values of multiple variables) may become neighbours in the feature representation, i.e., differ by a single higher-level feature.

Fig. 2 illustrates the corresponding effect to the fitness landscape by transforming the representation of a solution into a set of features. Relative to the feature (compressed) representation of a solution, fitness differences will be due to differences between features (unlike the original representation level where differences in individual variables cause fitness differences). Note, whilst the illustration shows a transformation of the fitness landscape, DO is not learning the fitness landscape, nor a compressed representation of the fitness landscape, nor learning movements that are good (like RL methods). DO is learning a compression of the variation, which corresponds to the reorganisation of the neighbourhood of a solution and, consequently, the fitness landscape organisation.

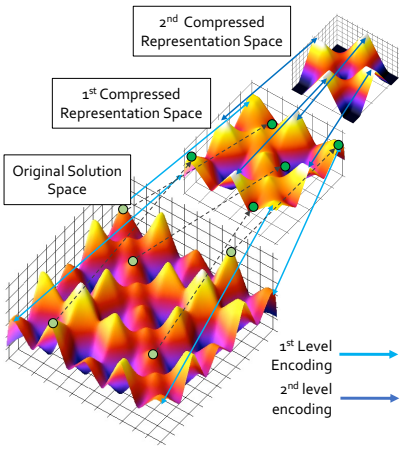
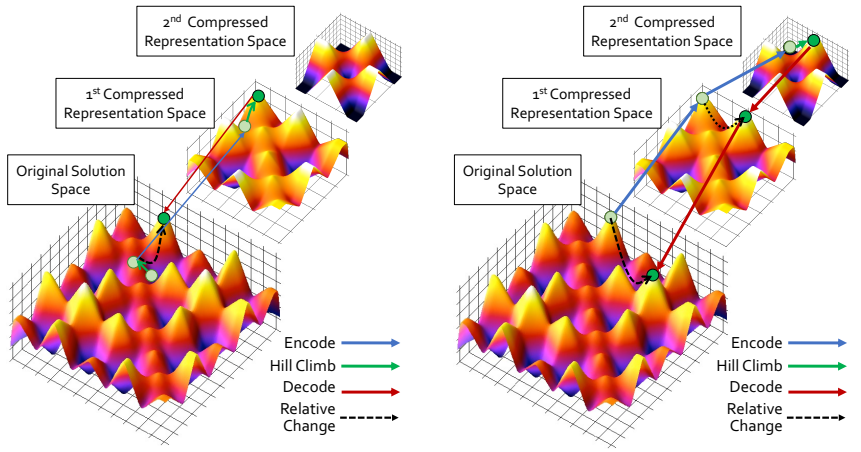


Fig. 2 Learning a compressed representation of the solutions found at the lower-level organisation induces a higher-order search space. Solutions that were originally far apart appear closer together in the reorganised space. The induced space contains macro-scale local optima where local search in the compressed solution space can also become trapped at sub-optimal solutions. This space, in turn, can be compressed, constructing a multi-level representation of the solution space

3.3 Searching in the compressed representation:

The compressed representation allows for search to continue in a reorganised space. This is demonstrated in Fig. 3. A solution is first optimised in the original search space. Once at a local optimum, the solution is trapped, and hill-climbing can no longer be performed at this level of organisation. Next, the solution is encoded to the first compressed representation. At this level of organisation, hill-climbing can be resumed by changing individual features — local search at the compressed representation level. Specifically, a local variation is performed by making a change to a feature representation and then decoding back to the original level of organisation. Relative to the solution representation, the change appears as a large coordinated change to the original problem variables — yet at the higher-order representation, the change was only a local change. The higher-order variation enabled the solution to escape the local optimum and 'jump over' a fitness valley. Note that such a jump does not guarantee that the point arrived at, on the other side of a fitness valley, is a point of higher fitness. Rather it is useful because it can sample this distant point without visiting all the low fitness points in the valley between. Selection on this higher-level unit can then assess whether the change was an improvement or not. The changes thus enabled are thus not directed in the sense of knowing in advance whether they will afford an improvement - rather they are useful because they collapse the dimensionality of the exploration process (by using information learned from past experience to reduce the sampling of combinations that are rarely unfit). Consequently, at the compressed representation, hill-climbing can continue to find superior solutions that may otherwise be pathologically difficult to find at the original level of organisation (illustrated in Fig. 3(a)).



(a) Hill-climbing in the original space gets trapped at a local optimum. The solution is then encoded to the compressed representation, where hill-climbing can resume. The change in the compressed space is decoded as a large but coordinated change in the original space. Consequently, enabling search to escape the local optimum

(b) Hill-climbing in the deeper representation is required to escape the local optimum at the 1st compressed representation. This makes a large coordinated change in the 1st layer space and even larger coordinated change in the original space

Fig. 3 Hill climbing in a higher-order representation can result in a large and coordinated change in the original solution representation. This can be achieved by encoding a solution to the higher-order representation, making a local variation in this new space, and then decoding back to the original representation

3.4 Successive Transitions in Individuality

At the solution level, there exist many local optima that trap a local search operator. Searching at the compressed representation causes a higher-order variation that can escape these fine-scale local optima. In turn, searching in the compressed solution space can also become trapped, i.e., solutions that differ by a single higher-level feature do not improve the current solution, and therefore the problem contains macro-scale local optimal (Fig. 3(a)). To escape a local optimum at the macro-scale, an additional compression is learned. By encoding to the second layer of organisation, hill-climbing can again be performed. Illustrated in Fig. 3(b), the local change at the second level of organisation is decoded back, performing a large coordinated change to the first layer of organisation and an even larger but still coordinated change to original solution representation. Further, the idea of inducing a compressed representation can be performed indefinitely, constructing a multi-level compressed representation of the search space.

Each level of organisation induces a reorganised representation of a solution, which in turn transforms the neighbourhood, and therefore fitness landscape, of the solution space, as illustrated in Fig. 3(b). The evolutionary search process does not change; it is just rescaled into a new representation. Specifically, each solution is updated by hill-climbing at the compressed representation finding locally-optimal

solutions in the compressed representation. The distribution of solutions found at the compressed representation level will contain information about how the units of variation at the compressed representation level (higher-order units of variation) combine to improve the fitness of a solution. In the language analogy, the distribution of solutions can contain information about how syllables can combine into words. An additional compressed representation can be induced, compressing the units of variation at the first layer of organisation to an even higher-order unit of variation — providing large coordinated variation when decoded into the first layer of organisation.

The idea of search in reorganised neighbourhoods shares similarities with the well-known Variable Neighbourhood Search method [18], where a fixed search is performed using a number of different neighbourhoods. However, in VNS these neighbourhoods are manually defined in advance, whereas in DO, a compression of a neighbourhood is induced from solutions found in a larger neighbourhood, providing a new and intelligent space to search in without requiring *a priori* domain knowledge. Also, it has been shown that constraints in a mathematical programming formulation can be used to adapt the search space during optimisation, which in-turn has produced good results for challenging problems [47, 39]. Here, constraints are applied that break the symmetry that can appear in repeated runs, which in effect filters out and avoids redundant areas of the search space that have already been visited. This shares similarities with DO, and MBOAs in more general, where the reorganisation of the solution space is not defined prior to the run (unlike in VNS) and emerges due to the solutions found during optimisation. However, MBOAs differ to this approach as a machine learning model is used to adapt the search space rather than utilising predefined constraints. Further and more detailed parallels between DO's architecture and operation, and those of related methods, is the subject of the next section.

4 Comparing characteristics of Model-Building Optimisation Algorithms

Before running experimental comparisons, we first discuss how different features of various MBOA's might correspond to different problem-solving abilities. The characteristics of a fitness landscape present challenges that an algorithm must overcome to find a superior solution efficiently. We are interested in methods that explore the fitness landscape in order to find a globally optimal solution despite these challenges. MBOAs explore the landscape using processes inspired by natural evolution. The model captures relationships between variables from a distribution of promising solutions. These relationships are then exploited by informing future search. Among different state-of-the-art MBOAs, there exist substantial differences both in the type of models that are learned, and also in the way in which the models (once learned) are used to inform the search process. Consequently, we expect these differences to effect the performance of an MBOA. Further, we expect the introduction of a new MBOA that uses a deep neural network in a new way, providing a novel contribution to the class of MBOAs, to also have a significant impact on the adaptive capabilities. In this section, we introduce the state-of-the-art MBOAs, and also examine the differences between the algorithms more closely to separate the capability of DO from the state-of-the-art MBOAs.

4.1 Linkage Tree Genetic Algorithm (LTGA)

LTGA [46] uses agglomerative clustering to construct a hierarchical tree compression of the linkage information. The linkage information is provided by the dependency structure matrix (DSM), representing the variation of information between two clusters, populated from a distribution of promising solutions. Each variable is initially considered a separate cluster. After each clustering step, the DSM is updated to include the clustering. The outcome is a tree data-structure of linkage-sets, with each set representing a compression of lower-order linkage-sets.

New solutions are found by using a search method called optimal-mixing: As a generalised analogue of crossover in sexual recombination, the constructed linkage-set determines which variables to exchange between solutions. The model thus represents the structure of dependencies between variables, not the values assigned to variables. Values are constructed from a random solution drawn from the population. As such, the variation applied to a solution is dependent on the population and linkage-set. Each linkage-set is utilised by traversing the tree with each beneficial exchange being kept. This is applied to all solutions in a population. A new model is then constructed using the new distribution of solutions.

4.2 Parameterless Population Pyramid (P3)

P3 [15] is an advancement of the LTGA, that uses multiple incremental tree linkage-sets as the model. P3 maintains multiple populations arranged in a hierarchy. Each level of the hierarchy can be summarised as an LTGA instance, and thus each population has its own linkage-tree model that exploits information contained only at the corresponding population level. What differs significantly from other state-of-the-art MBOAs is how the populations are managed. A solution is generated one at a time. A local search is applied to the solution to provide a solution containing variable combinations that contribute to the solution quality. This solution is then added to the lowest level population that does not already include this solution. When a new solution is added to a population, the linkage-tree model for that population is reconstructed. The solution is then update via optimal mixing using the population and model only at the current level in the population hierarchy. If the solution is improved, the solution is added to the population at the next level in the hierarchy. If no improvement is found, then the solution is left in the population, and the algorithm restarts with a new solution. The significant advantage of P3 is that it requires no tuning of the population size. However, the algorithm exploits a type of incremental learning — an online learning method — to update the model, which can become costly as many models can be reconstructed at each iteration.

4.3 Hierarchical Bayesian Optimisation Algorithm (hBOA)

hBOA [34] uses a Bayesian network to represent variable dependencies in a distribution of promising solutions. The construction process uses a greedy algorithm that

adds directed edges to an empty graph based on how much it improves the Bayesian information criterion — a pairwise metric that accounts for the likelihood function and model complexity. The model construction requires both learning the linkage structure and conditional probabilities. Learning high-order interactions causes an exponential increase in the number of parameters. hBOA limits this by representing regularities in conditional probabilities using decision trees.

New solutions are generated by sampling the model to generate a complete solution (MIG). Restricted tournament replacement (RTR) is then used for solution replacement. Specifically, the solution, within a subset of the population, that is the nearest neighbour (Hamming distance) to the generated solution is used for competition. The solution with greater fitness is retained. As such, this process is functionally related to searching in a redefined neighbourhood. MIG is repeated to generate a new distribution of solutions. A new model is then constructed using the new distribution of solutions.

4.4 Dependency Structure Matrix Genetic Algorithm (DSMGA-II)

DSMGA-II [22] uses an incremental graph linkage-set as the model. DSMGA-II maintains a population of candidate solutions. The model is constructed from the distribution of solutions selected using binary tournament selection. New solutions are generated by two extensions of optimal mixing (restricted-mixing and back-mixing). However, the method of finding new solutions shares the same idea as LTGA and P3, where the linkage-set is used to determine which variables exchange states between solutions. The subtle difference comes from deciding which solutions to use for the exchange. Like LTGA, the linkage set is constructed using a DSM. However, instead of using agglomerate clustering, DSMGA-II constructs a linkage set by searching for a specific sub-graph called the approximation maximum-weight connected sub-graph (AMWCS). The linkage set produced is a graph structure: it is possible for a cluster to have multiple parents.

4.5 Deep Optimisation (DO)

DO [6] constructs a deep autoencoder model using a layerwise procedure. The autoencoder model consists of an encoder (E) and decoder (D) network that transforms an input to a latent representation (H) and then back to the original input representation. Specifically, $S_r = D(E(S))$, where S and S_r is a solution and solution reconstruction respectively.

The algorithm is presented in Algorithm 1. A population is first initialised. Local variation and selection in the solution representation are applied (to a set of solutions independently), generating a distribution of promising candidate solutions. An autoencoder with a single hidden layer is then trained using the distribution of solutions as the training set. The parameters of the encoder and decoder are updated using back-propagation to minimise the error between the input solution and reconstructed solution. Dropout is used during training, to encourage a latent representation that captures the relationships between the variables.

Algorithm 1: Deep Optimisation

```
Initialize: Population of Solutions;
Initialize: Single layered Autoencoder model;
T=0;
while T < Maximum Number of Transitions do
    for Solution in Population do
        while Optimizing Solution do
            Apply variation and selection to Solution using MIV
    // apply transition
    if T > 1 AND Model Depth < Maximum Model Depth then
        Add another hidden layer to the autoencoder model;
    Train autoencoder using the population as training set;
    T=T+1;
```

Search then continues at the new latent representation using the learned model this is the first 'transition' and is illustrated in Fig. 4. Specifically, each solution in the population is updated in the following manner. A solution is encoded to produce a latent representation, H. A local change is then made to the latent representation, producing H'. Both representations are then decoded and binarized, to produce S_r and S'_r respectively. A new model-informed variant, S', is constructed from S using $S' = S + (S'_r - S_r)$. Intuitively, this approximates $S' = S'_r$ but avoids the requirement that the autoencoder reconstruction is perfect. If S' is fitter than S then S' is kept; otherwise, S is retained. MIV of this form is iterated to accumulate further improvements if available. We refer to this as local search in the latent space. When applied to all solutions in the population this produces a distribution of solutions that are locally optimal relative to neighbourhood of the latent representation. A new hidden layer is then added to the model and trained, updating the parameters for both layers, using the new distribution of solutions. This is the second transition. This process is repeated through multiple transitions, applying local search in the latent representation of solutions and then training a new layer added on to the autoencoder. When the model has multiple layers, MIV can be applied by perturbing a variable at any layer. In this paper, MIV first searches at the deepest layer of the autoencoder model for a solution, and then searches at the solution level. DO, as the other MBOAs, performs best by prioritising simpler models over more complex models. In hBOA and LTGA this parsimony pressure is implicit in the model construction. DO enables this to be explicitly controlled using standard neural network techniques such as L1 and L2 regularisation.

4.6 How do these methods vary in their Model Induction and Model-Informed Search capabilities?

The algorithms introduced above have many complex features and differences one from the other. This makes it rather difficult to understand what one algorithm might be able to do that another cannot. Here we examine these differences more carefully in terms of their consequences for the structure they can represent, their methods to

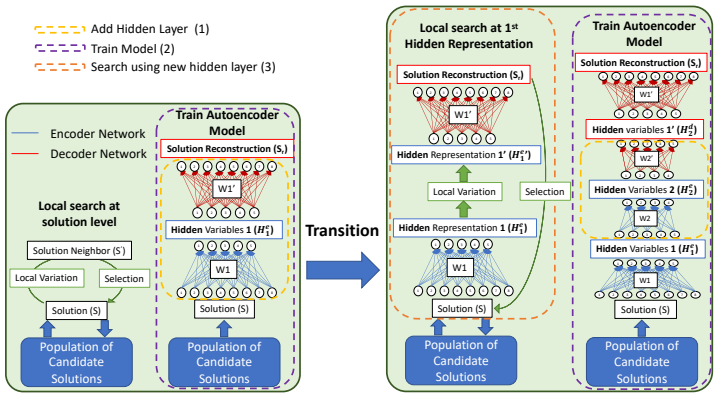


Fig. 4 Schematic of the transition process performed by DO. First, the population of solutions is updated by performing local search at the current level of organisation (initially, the solution representation $L = 0$). A hidden layer, H_1, is then added to the autoencoder model (1). The autoencoder model is then trained using the distribution of solutions found (2). After training, a transition occurs, and each solution is updated by performing local search at the hidden representation (H_1) of the solution (3) - a process we call Model-Informed Variation (MIV). The process repeats: finding a distribution of locally optimal solutions relative to the deepest layer, adding a further hidden layer and updating the model, and then performing local search at the new hidden layer to update the population solutions.

Method	Properties								
	MIG	MIC	MIV	Pair-wise	Multi-v'te	Tree-based	Graph-based	Deep	'Houpi'
DO	×	×	✓	×	✓	×	✓	✓	✓
DO_1	×	×	✓	×	✓	×	✓	×	✓
LTGA	×	✓	×	✓	×	✓	×	×	×
hBOA	✓	×	×	✓	×	×	✓	×	×
DSMGA-II	×	✓	×	✓	×	×	✓	×	×
P3	×	✓	×	✓	×	✓	×	✓	×

Table 1 Summary of the key properties of the model-based optimisation algorithms considered in our experiments, where 'Houpi' means that higher order units (of variation) are population independent

induce this structure and how they use this structure to move in solution space. Table 1 summarises the differences between MBOAs. We make the following hypothesis about the types of problems that will distinguish their different capabilities.

1. LTGA and P3 use a strict binary tree to represent the dependencies between variables and are therefore restricted to representing non-overlapping dependencies. For DO, hBOA and DSMGA-II, this is not the case. Consequently, when the set of adaptive units of variation have overlapping solution variables, LTGA and P3 will necessarily fail to improve a solution, whereas DO, hBOA and DSMGA-II will not.
2. Model-Informed Generation (MIG) and Model-Informed Crossover (MIC) use a population of solutions to update a solution that is considered a neighbour in the compressed search space learned by the model. Model-Informed Variation (MIV) on the other hand does not use a population as the model contains all the necessary information. Therefore, if search removes the diversity in a population,

MBOAs using MIG or MIC (LTGA, P3, DSMGA-II and hBOA) will necessarily fail to find this fitter solution, were as MIV (performed by DO) will not.
3. All MBOAs, at some point, use pairwise statistics between variables to construct the model, whereas DO does not. Consequently, when the set of adaptive directions of variation appear independent when measured using pairwise statistics (or cannot be well approximated using pairwise statistics), all MBOAs will necessarily fail to search in the higher-order organisation, whereas DO will not.

The first hypothesis refers to overlapping dependencies which was identified as an issue early during the development of EDAs [3, 35]. However, benchmarks containing overlap used to demonstrate a multivariate models' performance turned out to be solvable in polynomial time using LTGA and P3 [16, 22, 8]. Therefore, an important distinction we make here is that we specifically refer to overlap in the set of adaptive directions of variation rather than dependencies in a fitness function.

The second hypothesis refers to the differences in methods used by MBOAs to construct a change. The methods are Model-Informed Generation (MIG), Model-Informed Crossover (MIC), and Model-Informed Variation (MIV). For a binary problem, suppose that in a distribution of promising solutions we observe that two particular solution variables frequently take the same value as each other i.e., 00 or 11. And suppose we learn a model that represents this relationship. In MIG, the model is used to generate 00 and 11 more often than 01 or 10. In MIC, the model represents variables v_1 and v_2 as a linkage-set. Crossover is then performed between two random solutions using the linkage set as a crossover mask. Considering the model captured this information from the same distribution of solutions used to perform crossover, the crossover operation is likely to exchange between values 11 or 00. In MIV, the model compresses and represents the factor of variation as a single dimension, i.e., the model represents 00 and 11 as neighbours even though they are not neighbours in the original space. The single-unit change is applied to the compressed representation unit, decoding back to the original problem variables as a simultaneous change to multiple solution variables. MIG is not pre-conditioned to account for the solution it is to replace or update. hBOA, however, uses Restricted Tournament Replacement (RTR), after MIG, that subsequently approximates the process of updating a solution that is local in the compressed solution space. Specifically, RTR finds a solution in the current population that is the minimum Hamming distance away from the generated solution. Selection is then applied, updating the solution if the generated solution is fitter. Therefore, MIG (+RTR) and MIC are limited approximations of local search in the neighbourhood defined by the model, where both methods rely on a population of solutions to update a single solution with a neighbouring solution in the compressed neighbourhood. MIV on the other hand does not require a population. Consequently, if search removes diversity in the population, MIG and MIC can be susceptible to failure whereas MIV will be less so.

Finally, the third hypothesis refers to the model construction methods used by all MBOAs. All models are constructed by adding linkage between nodes that show greatest measure of dependency in the distribution of solutions. However, as the order of the statistics increases the complexity of calculating the dependency information increases exponentially. Therefore, to simplify model construction, all MBOAs ex-

cluding DO use pairwise statistics to approximate the dependency information. DO, on the other hand, learns the parameter values by incrementally updating the weights of the network in the direction that reduces the reconstruction error. Therefore DO is not limited to pairwise statistics. Consequently, we hypothesise that DO will be capable of inducing a higher-order representation of variation when the information about how lower-level units interact cannot be measured using pairwise statistics. The characteristic of pairwise independent functions is known to cause failure to MBOAs that construct models using pairwise statistics [26].

In all discussed methods, an individual solution is perturbed using a centralised method (a single model that is used to update all solutions). The set of all possible adaptive changes for any given solution must therefore be captured and represented by the model. Which change is adaptive is dependent on the solution that is being updated. Therefore, the challenges an MBOA must overcome can be attributed to distribution of changes that are required to update the population of solutions. Therefore, the set of all variations — precisely the challenge of separating and representing the directions of variation — will directly affect the model induction and exploitation capability of an MBOA. In Section 6, we explore how optimisation problems can explicitly create the challenges described in these hypotheses and then evaluate each MBOA's and DO's performance to overcome these challenges. Before that, we provide a more general motivation of DO's problem-solving strengths by considering a well-known NP-hard optimization class: multiple knapsack.

5 Multi-Dimensional Knapsack Problem (MKP)

In this section, we evaluate the performance of MBOAs and DO using a well-known applied problem, with the focus on considering relationships between problem structure, and the ability of algorithms to exploit that structure. We use benchmark MKP instances from [9]. Results for a simple genetic algorithm (GA), LTGA and dBOA (hBOA = dBOA + RTR) are provided [27]. Here we compare these against results for DO and a variant of DO used as a control. Specifically, because DO has many differences from the other methods we want to test whether deep representations are really improving the problem solving ability, or whether it is the other algorithmic differences that are doing the work. To do this, we use DO_1 which is DO limited to use only one hidden layer. This control also enables us to assess whether the problems we are working on have structure that can be exploited by deep models that cannot be exploited by shallow models. Further, we include results found by a single-bit local search (LS) and a 2-bit local search (2bLS), allowing for item swaps, to illustrate the performance improvement made by the models.

The objective of MKP is to assign a set of items that maximises the combined value while within the m knapsack dimensions. Formally, MKP is expressed as

$$\text{maximize} \quad \sum_{j=1}^{N} p_j x_j \quad , \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{N} w_{ji} x_j \leq c_i \ , \ \ i = 1, \ldots, m \quad , \tag{2}$$

$$x_j \in \{0,1\} \ , \quad j = 1, \ldots, N \quad , \tag{3}$$

where p_j is the value of item j, N is the number of available items, x_j is a binary assignment determining if item j is selected, w_{ij} is the size of item j in dimension i and c_i is the total capacity of the knapsack in dimension i. The instances were constructed in the following way. The dimension size for each item w_{ij} was generated from a discrete uniform distribution $U(0,1000)$. The capacity of each dimension c_i was calculated as $c_i = \alpha \sum_{j=1}^{N} w_{ij}$ where α is called the tightness ratio. Instances with a smaller tightness ratio and or higher knapsack dimension are generally considered more complex. Comparison is made using an instance size of N=100, knapsack dimensions (m) of 5, 10 and 30 and tightness ratios (α) of 0.25 and 0.75.

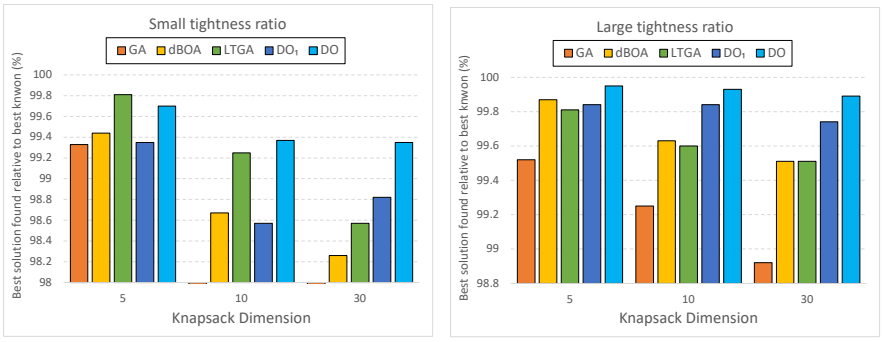
All algorithms, except DO, use a repair operator to overcome the problem challenge of infeasible solutions. Given an infeasible solution, the repair operator iteratively removes individual items, in the order of lowest to highest utility, calculated by $u_j = p_j / (\sum_{i=1}^{m} r_{ij})$, until no constraints are violated. Then, items are iteratively added, in the order of highest to lowest utility, to the solution until an item addition violates a constraint. When this occurs, the item is not assigned and the repair is terminated. DO does not use a repair operator; if a variation to a solution violates a constraint, it is rejected rather than repaired. DO therefore only stays within the feasible region of the solution space. Comparing DO with MBOAs that utilise a domain-specific repair operator puts DO at a significant disadvantage, and serves to demonstrate both the applicability of DO (without using methods specific to this domain) and its ability to exploit problem structure in applied problems.

All algorithms use a population size of 1000 and run until the population converges. The population of dBOA and LTGA are initialised using LS to improve the signal of good variable combinations. For DO, LS and 2bLS all solution variables are initialised with 0s (an empty knapsack). The average best solution gap found for 10 instances of a problem type m, α are reported in Table 2 and plotted in Fig. 5.

The comparison between LS and 2bLS shows a significant improvement when a variation operator can swap items in and out of the knapsack rather than only add items. This indicates the usefulness of domain-specific operators. The simple GA using a repair operator outperforms LS. hBOA, LTGA and DO all find superior solutions compared to the basic methods. Significantly, the results show that DO appears to provide good results compared to other MBOAs, with greatest performance observed in the most complex cases (large m). This shows that DO is able to exploit more structure from the population than other MBOAs. Further, by comparing DO with DO_1, the results show that a deep model is required to exploit this structure. The results presented here are a comparison of the models used in MBOAs. Whilst

Table 2 Performance evaluation on MKP instances

		% Gap from Optimum Fitness						
m	α	LS	2bLS	GA	dBOA	LTGA	DO_1	DO
5	0.25	14.69	6.79	0.67	0.56	**0.19**	0.65	0.30
10	0.25	15.44	6.03	NA	1.33	0.75	1.43	**0.63**
30	0.25	14.91	4.70	NA	1.74	1.43	1.18	**0.65**
5	0.75	5.02	1.69	0.48	0.13	0.19	0.16	**0.05**
10	0.75	5.66	1.42	0.75	0.37	0.40	0.16	**0.07**
30	0.75	6.03	0.84	1.08	0.49	0.49	0.26	**0.11**
	Model	NA	NA	NA	Bayesian Network	Linkage Tree	AE 1D	AE 6D



(a) Small tightness ratio ($\alpha = 0.25$) (b) Large tightness ratio ($\alpha = 0.75$)

Fig. 5 The average difference between the best solution and best known solution

improvements can be made to LTGA, hBOA and DO in the form of utilising domain-specific methods [28], for these results, the model is the primary method (or only method in the case of DO) that improves a candidate solution. Thus, an MBOA's performance can, we suggest, be improved by using a deep model.

These results demonstrate that DO is able to solve some problems better than other state-of-the-art methods. But we want to know why. What exactly is it that DO can do that the other methods cannot? Our main contribution, therefore, is to provide this understanding by using a synthetic problem construction to explore different problem challenges as per the hypotheses above.

6 Exploring characteristics of problem difficulty

In this section, we investigate the types of problem structure that distinguish the capabilities of MBOAs, including DO. Specifically, we identify problem characteristics that cause a polynomial vs exponential time complexity differentiation between the MBOAs. Previous works comparing MBOAs [46, 22, 16] have not provided examples that show such a distinction.

To clarify the differences between MBOAs we divide the issues involved into two: the type of models they use, and how they use those models to enhance search. That is, the differences between the type of neighbourhood compression a model can perform (differences in model capacity and construction) and the differences between the methods used to exploit information from the model to inform search. We do this because an MBOA may be capable of accurately reorganising a solution's neighbourhood (sufficient model capacity), but cannot efficiently exploit the information to explore it. Conversely, an MBOA may be capable of efficiently exploring the neighbourhood of a solution if it were given a suitable model, but the method cannot learn such a model. To explore these capabilities separately, we use a construction for synthetic optimisation problems that separates the complexity of reorganising a solution's neighbourhood (model induction) and the type of search to perform in the reorganised neighbourhood (model exploitation). We achieve this by separating the fitness function, that normally maps a solution S directly to fitness, into two parts: a compression mapping C, that maps S into a higher-level binary representation R, and an environmental mapping (E) that maps R into fitness; detailed in Equation 4,

$$F(S) = E(R) = E(C(S)) \quad . \tag{4}$$

In this way, C is used to control how difficult it is to induce a compressed representation and E is used to control how difficult it to find higher-order solutions given that compressed representation.

What makes this problem construction useful is that the change at S, informed by the model of an MBOA, required to make a single-unit change at R can be defined independently from how these changes are used to find higher-order solutions. This problem construction enables us to attribute the performance differences between MBOAs to specific problem characteristics related to either the reorganisation of the neighbourhood of a solution, by changing C, or the ability to explore the reorganised solution neighbourhood, by changing E. In the sections that follow we develop three different compression mappings and three different environment mappings, and then we test the algorithms on the 9 combinations of these mappings. Each mapping is constructed to test a different capabilities of MBOAs. We begin with relatively simple aspects of difficulty (such as simple modularity that requires an algorithm to generate such modules). We then progress through more difficult properties that require either more advanced model induction or more sophisticated search using the model. We empirically find that some of these properties are more difficult than others insomuch as fewer of the standard algorithms are able to cope with these problem characteristics. Accordingly, in a rough sense, we end up with 'easy', 'medium' and 'hard' versions of each mapping (Table 3).

The resulting range of problems is thereby carefully developed such that if an algorithm can overcome the problem characteristics, finding a global optimum takes polynomial time; otherwise, the algorithm takes exponential time. This allows for a scaling analysis to identify problem characteristics that an MBOA can do (polynomial running time) and problem characteristics that an MBOA cannot do (exponential running time).

In introducing this synthetic problem, some unfamiliar terminology is introduced. Table 3 provides a glossary of acronyms used.

Term	Description
R	Higher-level binary representation
C	Compression mapping: maps from solution space, S, to R
E	Environmental mapping: maps from R to fitness
BB	Building block: a subset of solution variables
PS	Partial solution: values assigned to a BB
Compression mappings	
NOV	Non-overlapping variation (easiest case)
OV	Overlapping variation (medium difficulty)
NPOV	Non-Pairwise overlapping variation (hardest case)
Environment mappings	
GC	Generating combinations (easiest case)
HGC	Hierarchically generating combinations (medium difficulty)
RS	Rescaling search (hardest case)

Table 3 Glossary of acronyms and terms used in Section 6

6.1 The Compression mapping (C) from solution space to new representation

The Compression Mapping (C) defines the relationships an MBOA model must learn and represent in order to efficiently search at the higher-level representation R and consequently follow the fitness signal. An MBOA can only apply variation to the solution representation S. Therefore, in order to move efficiently at R, the MBOA must induce the relationships in C such that model-informed search applies the correct change at S to move at R.

To ensure that C is learnable as the problem size increases, we construct a compression map using a set of building-blocks (BBs) — a familiar construction used for evaluating the performance of MBOAs (technically, these modules are not building blocks in the sense of the building-block hypothesis familiar in genetic algorithms because they do not have tight linkage [14]). Each BB maps a disjoint set of solution variables (S_m) to representation variables R_m as detailed in Equation 5,

$$R = C(S) = R_1, \ldots, R_m = bb(S_1), \ldots, bb(S_m) \quad , \tag{5}$$

where bb is a BB mapping and m is the number of BBs. A BB performing a compression from four solution units to two binary representation units is sufficient for the purposes of this paper, i.e., to distinguish the model induction abilities of all the algorithms (compression's to one unit can be solved using a simple model [23]). The compression (two binary units in R) identifies four particular configurations in S (each containing four solution bits) from the sixteen possible that we call partial-solutions (PS). Each PS is represented by a unique binary code at R_m in the two binary units of R. A Combination in S_m that is not a PS is represented at R_m by null values. The E map only rewards non-null values in R. Therefore, in finding a PS to a BB, the PS must be substituted with an alternative PS to avoid deleterious fitness changes.

The PS set controls the compression complexity of a BB. We synthetically determine this to induce particular characteristics in the higher-order unit substitutions required in S to vary between PSs (i.e., the change required to move to an alternative PS in a single step). Note, there exist other operators to move between PSs, we specifically refer to the substitutions that allows movements from a PS to its near-

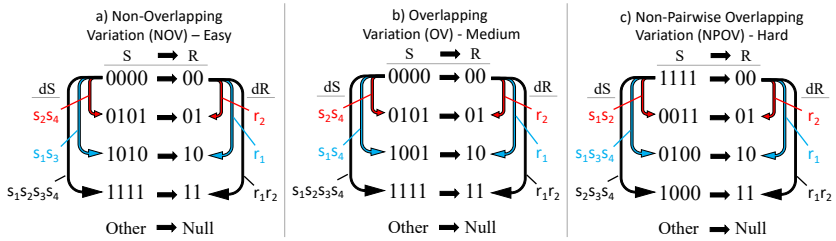


Fig. 6 A BB of size four contains four partial solutions each represented by a unique binary code at *R*. Alternative combinations are represented by nulls. The set of partial solutions controls the multi-unit substitution, ΔS, an MBOA needs to perform to make a single-unit change in *R*. The characteristics of multi-unit substitutions explored are (a) non-overlapping, (b) overlapping and (c) non-pairwise overlapping

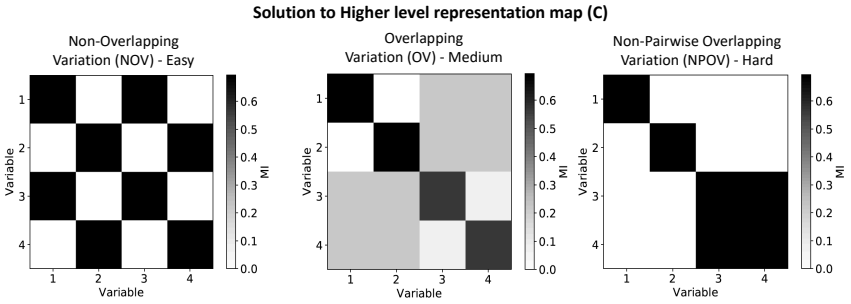


Fig. 7 The mutual information within a BB for each compression mapping (*C*) type

est PS using a single unit change, i.e., a higher-order unit substitution that makes a single-unit (local) change in *R*. The types explored in this paper are Non-Overlapping Variation (NOV), Overlapping Variation (OV) and Non-Pairwise Overlapping Variation (NPOV), which are considered easy, medium and hard difficulty respectively. Fig. 6 illustrates, for each variation set type (BB mapping), the change required in *S*, ΔS, to make a change in *R*.

The NOV case (easy case) presents the baseline complexity where substitutions performed to move between PSs do not overlap. All MBOAs are capable of representing NOV, evidenced by [36, 16, 22]. The set of PSs: {0000,0101,1010,1111} create an instance of higher-order substitutions that do not overlap. Specifically, the single-point substitution (ΔS) that changes a PS to an alternative neighbouring PS (a hamming distance of 2 away) is accessed by a substitution in the set $\Delta S = \{\{s_1 s_3\},\{s_2 s_4\}\}$, where s_n represents that the value of variable *n* is changed during the substitution. In the case of NOV, each entry is disjoint from all other entries, i.e., a variable is one element of the substitution set.

The OV case (medium difficulty) present a complexity where higher-order substitutions do overlap. The set of PSs {0000,0101,1001,1111} is an instance that contains overlap in the set of higher-order substitutions. Specifically, the single-point substitution (ΔS) that changes a PS to an alternative neighbouring PS (a hamming distance of 2 away) is accessed by a substitution in the set $\Delta S = \{\{s_1 s_4\},\{s_2 s_4\},\{s_1 s_3\},\{s_2 s_3\}\}$.

In the case of OV, each entry is not disjoint, i.e, a variable can be in multiple elements of the substitution set.

In the example illustrated in Fig. 6.b, for PS 0000, the single-point substitutions that change the PS to a neighbouring PS share the variable s_4. For example, improving S, in the case of OV, the substitutions that changes S to an alternative PS is $\{s_1,s_4\}$ or $\{s_2,s_4\}$, where s_4 is shared. For NOV (Fig. 6.a), the substitutions that changes S to an alternative PS is $\{s_1,s_3\}$ or $\{s_2,s_4\}$, where no variable is shared.

Finally, the NPOV case (hard case) contains an overlapping complexity where the linkage cannot be identified using pairwise statistics. Specifically, the differences between solutions, and consequently the set of substitutions that moves between PSs, appear univariate using pairwise statistics — a property called pairwise independent functions [26]. The set of PSs, $\{1111,0011,0100,1000\}$ creates the instance that contains non-pairwise identifiable overlap. The single-point substitution (ΔS) that changes a PS to an alternative neighbouring PS (a hamming distance of 2 away) is accessed by a substitution in the set $\Delta S = \{\{s_1s_2\},\{s_1s_{3,4}\}\}$. The complexity is contained at variabless_1, s_2 and s_3, variable s_4 is used to maintain a consistent module size across module types and thus takes the same value as variable s_3 (does not add to the complexity).

Fig. 7 presents the mutual-information between a pair of variables for each BB type. In the case of C=NOV, linkage information does not overlap. For, C=OV, linkage information does overlap. For C=NPOV, variables s_1, s_2 and s_3s_4 appear independent yet two variables must change simultaneously to avoid a deleterious fitness effect. A problem instance uses the same BB mapping for all BB. Therefore, the complexity of C is attributed to the particular characteristics of a BB mapping, i.e., if an algorithm can learn a BB mapping, learning more of the same only increases the complexity of separating the building-blocks. The optimal PS to use in each BB is a function of the dependencies between building-blocks, controlled separately by E. The optimal PS to use in each module is a function of the dependencies that exist between the modules, which are defined by the environment mapping E.

6.2 The Environment mapping (E) from the new representation to fitness

The E map defines how PSs, and therefore units in R, interact to provide a higher-order solution (a combination of PSs). We explore different E mapping types that induce different search characteristics. The Hierarchically Generating Combinations (HGC) mapping requires an algorithm to rescale variation to higher orders of organisation repeatedly and is considered our medium difficulty case. The Rescaling Search (RS) mapping requires an algorithm to perform a local search in the reorganised representation and is considered our hard difficulty case. Finally, the Generating Combinations (GC) mapping is the baseline case, and therefore considered our easy difficulty case, where MIV, MIG and MIC can easily follow the fitness signal to higher-order solutions. The types of E mappings investigated here are relatively straightforward, i.e., a globally optimal solution can be found in polynomial time using a hill-climber or an MBOA when S=R (when C is an identity map).

A PS is easily found by rewarding only non-null values in R. Specifically, all E maps have the contribution defined in Equation 6,

$$F_r = \sum_{i=0}^{m} f(R_i),\ f(R_i) = \begin{cases} 0, & \text{if } R_i = \text{null} \\ 1, & \text{otherwise} \end{cases} . \tag{6}$$

Note, during optimisation, a MBOA compresses the search space. Consequently, it is important that during optimisation, the complexity in the compression mapping is not removed when combining combining PSs. This is achieved by ensuring that all PSs are required at all stages of the search process until a global optimum is found (i.e., the loss of the ability to access a PS will result in failure of the algorithm to find a globally optimal solution). This property is included in the synthetic optimisation problem by separating R into two sets, R_1 and R_2. Each set contains only a single representation unit from each BB; thus, each set contains m representation units (a compression of a module produces two representation units). Both sets are used to calculate the fitness of the solution and therefore, during search, the compression performed in R_1 is independent from the the compression performed in R_2, and vice versa. Consequently, a compression only occurs between modules and not within, ensuring all PSs are required during optimisation.

The baseline E mapping (easy case), called Generating Combinations (GC), requires a MBOA to only combine a PS one at a time with the PS that has the majority in the solution being optimised. Consequently, the signal that leads to a globally optimal solution can be identified from a distribution of randomly generated solutions that conserve PSs. Thus all MBOAs, regardless of the differences between their model-informed search methods, can efficiently follow a signal to higher-order solutions. Thus the mapping evaluates an MBOAs capability to learn C. The fitness (Equation 7), is a summation of the Hamming weight ($H()$) distance of each R set from the middle hamming weight of a subset ($m/4$). The separation of R_1 and R_2 produces a problem containing four global optima; each one containing the same PS in all BBs,

$$F = F_r + |H(R_1) - \frac{m}{4}| + |H(R_2) - \frac{m}{4}| \quad . \tag{7}$$

Interesting cases for MBOAs arise when a subset of low-order components can form high-order components, such as the case of hierarchical problem structure. Here we use the same hierarchical construction used in [52] as all MBOAs can overcome this characteristic when C is the identity map. This is because higher-order combinations can be efficiently identified by generating a distribution of solutions using the lower-level combinations. Therefore, this mapping evaluates an MBOA's capability to represent combinations of the variation operators at multiple scales of organisation. We name this mapping Hierarchically Generating Combinations (HGC) (medium difficulty). The hierarchical dependencies are represented by a binary tree containing $D = log_2(2m)$ layers. Each layer l contains 2^{D-l} parent nodes. Each node (r_l) represents an additive combination (A) of two child nodes (r^i_{l-1}, r^j_{l-1}) from the layer below. A parent node represents the common value if the child nodes contain the same value (excluding nulls), otherwise it represents a null. This compression is

provided in Equation 8. A parent node with a non-null value is assigned a fitness benefit. The total fitness, Equation 9, is the sum of all fitness contributions from all parent nodes,

$$r_{l+1}^{i} = A(r_l^i, r_l^j), \;\; A(r_i, r_j) = \begin{cases} r_i, & \text{if } r_i = r_j \\ \text{null}, & \text{otherwise} \end{cases}, \tag{8}$$

$$F = F_r + \sum_{l=0}^{D} \sum_{i=0}^{2^{D-l}} f(r_l^i), \;\; f(r_i) = \begin{cases} 0, & \text{if } r_i = \text{null} \\ 1, & \text{otherwise} \end{cases}. \tag{9}$$

The hierarchical linkage is shuffled for each instance to ensure that overlapping variation operators are present at all representation scales.

Fig. 8 illustrates how the depth of overlap can be controlled. The figure contains a problem of size N=16. Each node on the bottom layer is a solution variable, and each node on a higher level is a representation variable of the solution. The lines between nodes represent the linkage between nodes. At representation L_1, there are 4 representation nodes, colour coded to identify the dependent solution variables. Here a solution variables is a member of multiple representation nodes at L_1. At representation L_2, there are 2 representation nodes, numbered to identify the dependent solution variables. Here a solution variables is a member of only a single representation units at L_2. Thus the problem has been reduced from overlapping variation at representation L_1 to non-overlapping variation for L_2.

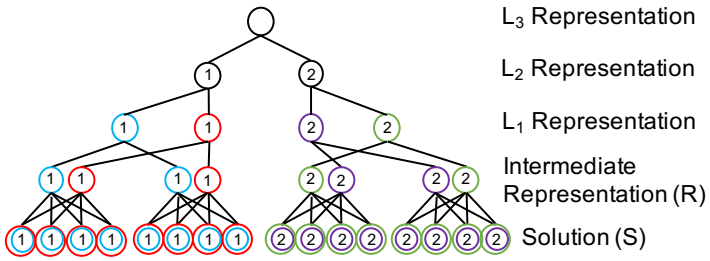


Fig. 8 E=HGC with overlap up to L_1 Representation. At L_2 Representation a change to a representation unit, e.g, unit 1, causes a change to solution units that are disjoint from other representation units at the same level, e.g., solution variables that are a member of representation unit 2. At the L_1 Representation, changes to representation units, e.g., colour red shares variables with other representation units, e.g. colour blue, at the same level. Therefore, L_1 Representation contains overlap and L_2 Representation does not contain overlap

.

The last type we explore is when higher-order components are not found by combining lower-order components. Rather they are found by searching in the space of lower-order components. Therefore, this mapping evaluates an MBOA's capability to exploit information from the model and search in the learnt representation (reorganised neighbourhood), hence we name this mapping Rescaling Search (RS) (hard

case). This characteristic is created by defining a unique path (UP) to the global solution. The location on the path is provided by coordinates: The Hamming weight of each R subset is used as coordinates for a 2D fitness mapping:

$$F = F_r + UP[H(R_1), H(R_2)] \quad . \tag{10}$$

The mapping contains a monotonically increasing slope function that takes any solution towards the start of the path at coordinates $(R_1, R_2) = (m, m/2)$. The path proceeds to coordinates $(m,0) \rightarrow (0,0) \rightarrow (0,m) \rightarrow (m,m)$, where (m,m) is the global optimum. Each step along the path increases the fitness, and thus deviations from the path cause a deleterious fitness. Using R_1 and R_2 ensures that local variations in R must be performed. A non-local change would cause a change to the other coordinate, causing a deviation from the path. Note, the path length scales polynomially with respect to the problem size and is easy to follow via local search in R, thus differentiating it from the long path problem [21].

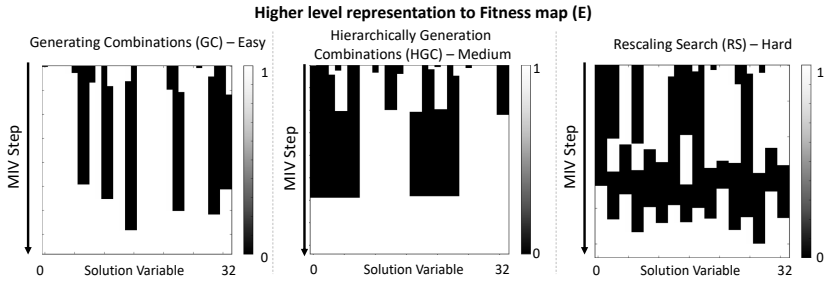


Fig. 9 An example of a solution trajectory, from a random solution (start of MIV) to a globally optimal solution (end of MIV) for each E map when C=NOV performed by MIV

Fig. 9 presents examples of solution trajectories, from a random solution initialisation (start of MIV steps) to a globally optimal solution (end of MIV steps) for each E map type. In these examples, C=NOV. For E=GC, search first identifies PSs and then continuous by substituting individual PS. In the case E=HGC, search is repeatedly rescaled to higher-orders of organisation, where the last adaptive variation makes a simultaneous change to half of the solution variables. For, E=RS, search cycles through multiple PS for each BB to find the global optimum. Note for E=RS all PSs are used to search for the global optimum.

The combination of maps C and E create a complex fitness landscape relative to S. The capability of an MBOA to reorganise the neighbourhood of a solution, by capturing relationships using the model, is evaluated by differences in the C map. The capability of an MBOA to exploit the information from the model, to explore the reorganised neighbourhood, is evaluated by differences in the E map. The synthetic construction ensures that if an MBOA does not accurately reorganise and search within the neighbourhood, it will take exponential time to find a globally optimal solution. Thus a scaling analysis will suitably demonstrate if an MBOA can or cannot overcome the problem characteristics. We have created three distinct types for

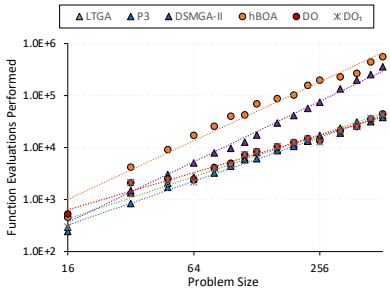
each mapping, namely: non-overlapping; overlapping and non-pairwise overlapping linkage information types for the *C* map, and generating combinations, hierarchically generating combinations and rescaling search types for the *E* map. For each type, we have created an instance of the characteristics, which we found to be sufficient to differentiate the performance of the MBOAs explored in this paper. In our experiments, we explore all nine combinations to investigate the performance differences between MBOAs.

7 Performance Evaluation

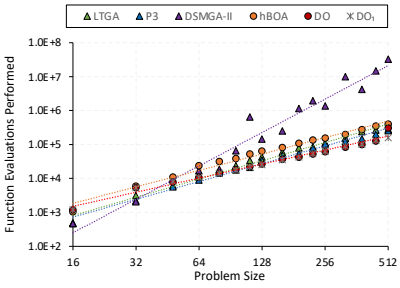
In this section, we assess the performance of LTGA, P3, hBOA, DSMGA-II and DO on all combinations of complexity and environment maps. Where appropriate, we also include results DO_l (The autoencoder model in Deep Optimisation is limited to l hidden layers). The performance of an algorithm is evaluated by performing a scalibility analysis. The results report the average number of function evaluations performed to find the global optimum solution in up to 10 independent runs (for runs with greater than 10^7 function evaluations, three independent runs are performed). We omit the computational cost of learning from our results as all learning methods incur a polynomial scaling and our investigating is to identify if an algorithm shows polynomial or exponential scaling. From this, we conclude if an algorithm can or cannot overcome the problem structure. Our results alone are not sufficient to claim which algorithm is showing greater performance when both algorithms show polynomial scaling. The population size is set such that within all independent runs, a global optimum is found. In the case of P3 no population size is set. The population is initialised using local search such that the distribution contained all PSs (removing potential challenges associated with finding the PSs). Algorithms are terminated if a global solution is not found within 10^9 function evaluations, and therefore no data point is provided. An advantage of LTGA and hBOA is that they do not have additional parameters to tune. A disadvantage is that this does not admit control over the inductive bias. DO, like other neural network methods, has several tuneable parameters. These values were selected from preliminary results on a small sample of problems. Specifically, hidden layer compression = 0.8 (maximum layers used was 9), dropout rate: 0.2, epochs: 400, learning rate = 0.002. Regularisation parameters in the range L1 = $[1\times10^{-3}:1\times10^{-5}]$, L2 = $[2\times10^{-3}:2\times10^{-6}]$.

7.1 Model Induction

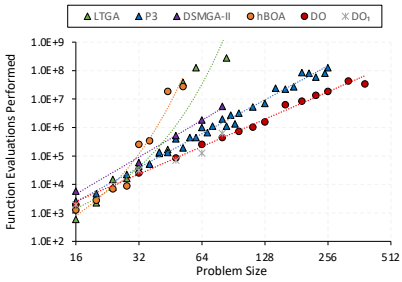
In this subsection we explore the model induction capabilities of the algorithms by keeping the environment mapping simple and varying the difficulty of the compression mapping (hypothesis 3). The Generation Combinations environment mapping (E=GC — easy case) is used to evaluate the performance of an MBOA to learn the C mapping types as all model-informed search methods can follow the fitness signal to a globally optimal solution. Fig. 10 presents the scalability performance of all MBOAs.



(a) *C*=NOV (easy case): All MBOAs show polynomial scaling



(b) *C*=OV (medium difficulty): All MBOAs show polynomial scaling



(c) *C*=NPOV (hard case): LTGA and hBOA show exponential scaling. DSMGA-II and DO_1 fail to find solutions for large problem sizes. DO and P3 show polynomial scaling

Fig. 10 Performance evaluation of an MBOAs model capacity to learn the different complexities in the neighbourhood reorganisation. All methods scale polynomialy when the model induction task is easy, but P3 and DO are the only methods to scale polynomially when the model induction task is complex.

For the non-Overlapping Variation mapping type (*C*=NOV), the easy case, all algorithms show a polynomial scaling (see Fig. 10(a)). We know that learning pairwise dependencies is straightforward for all models. Therefore, the results verify that all model-informed search methods are sufficient to find a globally optimal solution in polynomial time for *E*=GC if the model can learn *C*.

For Overlapping Variation mapping type (*C*=OV), the medium case, all algorithms show a polynomial scaling (see Fig. 10(b)). DSMGA-II shows a significant change in the performance compared to *C*=NOV. Notable is the change in the comparative performance of LTGA and P3 with DO and hBOA when compared to the *C*=NOV case. LTGA and P3 now show a performance more in line with hBOA. hBOA and DO, on the other hand, remained insensitive to the change in compression mapping, indicating that the overlap complexity affects the performance of LTGA and P3. However, as the degree of overlap is constrained to within modules (between modules, there are no overlapping relationships), the complexity is insufficient to cause exponential running times to LTGA nor P3.

For the non-Pairwise Overlapping Variation mapping type (C=NPOV), the hard case, DO and P3 are the only algorithms that scale polynomial (see Fig. 10(c)). As expected, the use of pairwise statistics limits the MBOAs ability to capture higher-order dependencies. It was unexpected to observe that P3 was successful. Recall P3 uses multiple linkage-tree models. We hypothesise P3's capability results from using a multi-level representation of the search distribution where the lower-level representations filter out a subset of the PSs and thus remove the complexity at higher representations. We also observe that DO_1 fails where as DO does not. This suggests that the higher-order relationships (greater than pairwise) required to learn the compression requires a deep model (greater than a single hidden layer).

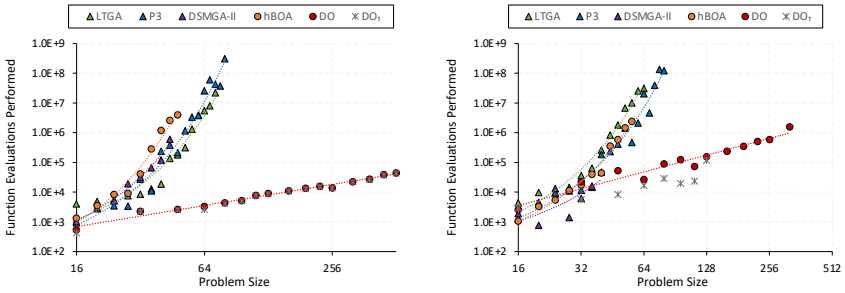
7.2 Model-Informed Search

In this subsection we explore the model-informed search capabilities of the algorithms (hypothesis 2). The Rescailing Search environment mapping (E=RS — hard case) is used to evaluate an MBOA to perform local search in the neighbourhood defined by the model. Fig. 11 presents the scalability performance of all MBOAs.

For all compression mapping types (see Fig. 11), only DO shows polynomial scaling. Therefore, Model-Informed Variation is the only model-informed search method that shows polynomial scaling. We know that all models are capable of learning the baseline compression complexity, C=NOV, (Fig. 10(a)). Therefore, the failure observed here is due to the model-informed search method and not the model induction complexity.

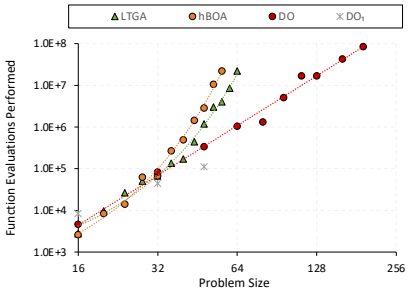
For C=NPOV mapping type (see Fig. 11(c)), we observe that DO_1 fails but DO does not. This further demonstrates that C=NPOV requires a deep model to represent a compression accurately. We have already seen that the MBOAs are unable to learn C=NPOV (evidenced by Fig. 10(c)). Here we see that MBOAs other than DO are unable to perform local search in the space represented by the model. Conseqently, this result shows that DO is overcoming the combined challenges that other MBOAs fail to do even when the problem challenges are separated.

To further demonstrate that E=RS presents a problem challenge that is easy for local search, we perform an experiment with C=I, where I is the identity mapping. Fig. 12 presents the results for all MBOAs and includes a hill-climber that performs single-bit substitutions to S. All algorithms do not use local at the representation of the solution (local searching using single-bit substitutions); otherwise this would not test an MBOAs capability. The hill-climber was able to find a global solution easily, along with DO. Note, here, DO must learn the identity function first and therefore is less efficient than the hill-climber. However, even when there is no complexity in the model induction, other MBOAs, that use MIG or MIC, fail. This result verifies that failures observed in the experiments with E=RS are due to how the model is used to inform search and not due to model-induction difficulty, supporting hypothesis two.



(a) *C*=NOV (easy case): Only DO shows polynomial scaling

(b) *C*=OV (medium difficulty): Only DO shows polynomial scaling



(c) *C*=NPOV (hard case): Only DO shows polynomial scaling

Fig. 11 Performance evaluation of an MBOAs capability to exploit information from the model to inform search. All MBOAs fail even when the model induction task is easy where as DO is the method that shows polynomial scaling. This failure can be directly attributed to model-informed search methods

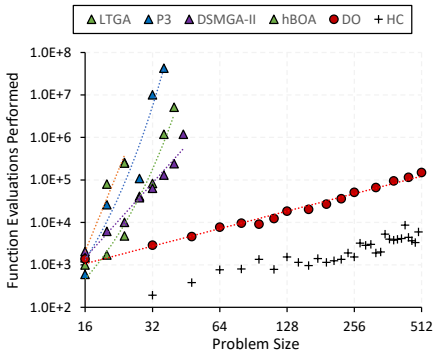
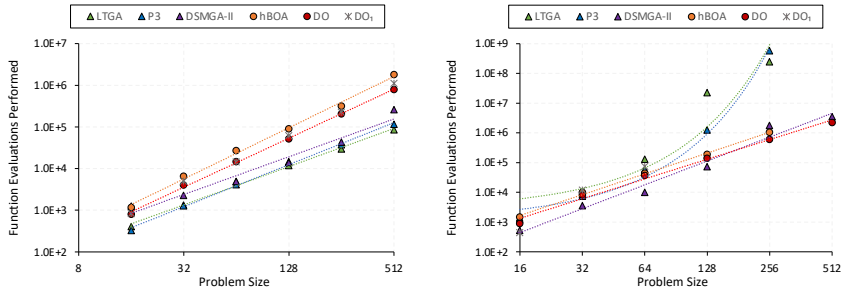


Fig. 12 Local search outperforms MBOAs that use MIG or MIC

7.3 Multi-Level Representation

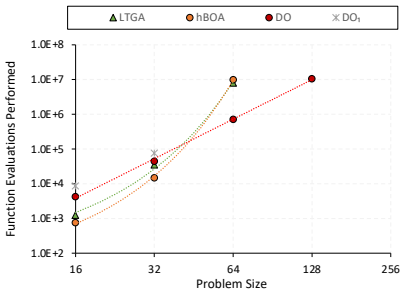
In this subsection we explore the multi-level representation capabilities of the algorithm (hypothesis 1). The Hierarchical Generating Combinations environment map-

ping (*E*=HGC — medium case) is used to evaluate the performance of an MBOA to combine variation operators to higher-orders of organisation recursively. The linkage information in *E* is shuffled such that overlap occurs between building blocks at all scales of organisation in the hierarchy. Fig. 10 presents the scalability performance of all MBOAs.



(a) *C*=NOV (easy case): All MBOAs show polynomial scaling

(b) *C*=OV (medium difficulty): LTGA and P3 show exponential scaling



(c) *C*=NPOV (hard case): Only DO shows polynomial scaling

Fig. 13 Performance evaluation of an MBOAs capability to recursively reorganise a solutions neighborhood to higher-orders of organisation. LTGA and P3 show exponential scaling in case of overlap (*C*=OV). Only DO shows polynomial scaling in the case of non-pairwise overlap *C*=NPOV. MBOAs that uses models limited to tree data-structures fail to overcome overlap where as more sophisticated models do not

For the Non-Overlapping Variation mapping type (*C*=NOV), all algorithms show a polynomial scaling (see Fig. 13(a)). We know that pairwise learning dependencies are straightforward for all algorithms. Therefore, the results verify that all algorithms can recursively compress a solutions neighbourhood and all model-informed search methods are sufficient to find a global optimum.

For Overlapping Variation mapping type (*C*=OV), LTGA and P3 show exponential scaling, whereas hBOA, DSMGA-II and DO show polynomial scaling (see Fig. 13(b)). The presence of overlapping variation operators is sufficient to cause methods that use tree-based models to fail. This result is because searching at higher layers of the hierarchical representation requires large variation, and in the case of *C*=OV, these

large variations contain a complex overlapping structure due to the interactions with other large variations, as illustrated in Fig. 8. This is supported by Fig. 10(b) where small overlapping functions did not cause failure to LTGA and P3. This result is the first time that demonstrates a problem type that hBOA can solve that LTGA and P3 cannot, supporting hypothesis one. Further, and as expected from the construction of the problem, DO_1 fails because it cannot represent deep problem structure whereas DO succeeds. The effect of deep representation is further explored in section 7.3.1 by performing experiments with a controlled depth of the overlap.

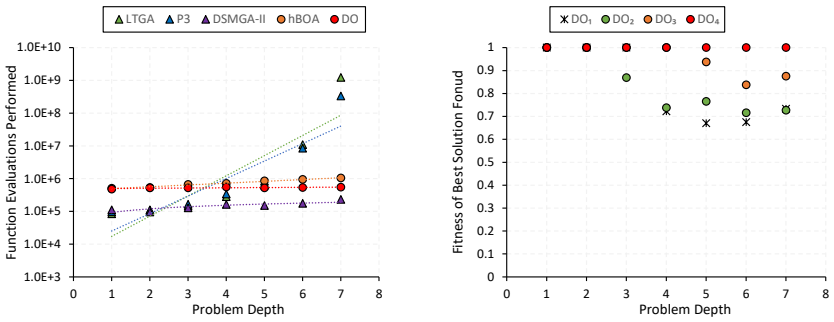
For the Non-Pairwise Overlapping Variation mapping type (C=NPOV), DO is the only algorithm to show polynomial scaling (see Fig. 10(c)). hBOA and LTGA fail due the inability to learn C=NPOV as evidence by Fig. 10(c). Results for P3 and DSMGA-II were not obtained, However, we also now understand that LTGA and P3 would also fail when it is necessary to recursively combine variation operators including overlap Fig. 10(c) and therefore expect P3 to fail in this case. We also expect DSMGA-II to fail due to its failure on the more straight-forward environment mapping, E=GC.

7.3.1 Problem Depth

During our experiments, we found that the size of overlapping variation operators (due to the hierarchical problem structure) caused significant challenges for MBOAs. In this section, we perform experiments that control the depth of overlap in the E=HGC environment mapping case to further our understanding.

The depth of overlapping BBs can be controlled by limiting the layer at which linkage is shuffled in the hierarchical structure of E=HGC. Thus, we can explore how the depth of overlap challenges these algorithms (whilst keeping the problem size constant). We perform experiments using the C=OV complexity for a problem size 256 and change the depth, d, at which linkage in E is shuffled. At layers greater than d no overlap is introduced (Refer to Fig. 8). Thus d controls the maximum size of an overlapping variation operator an MBOA needs to perform to find a global optimum. Further, we include results for when DO is limited to shallow representations where the depth of the autoencoder is limited to L hidden layers, denoted DO_L. We expect that deeper autoencoders (larger L) will be required to solve problems with deeper overlap (larger d).

Fig. 14(a) shows that LTGA and P3 are sensitive to the size of the overlap in the problem. Specifically, as the depth of the overlap increases the number of function evaluations required to find a global optimum increases significantly. DSMGA-II, HBOA and DO show no significant sensitivity. This demonstrates that the complexity of overlap is not a challenge for the model induction methods of these algorithms. Finally, we show that representing this complexity using the autoencoder model requires a deep model. Fig. 14(b) shows the fitness of the best solution found by DO in its normal operation (DO) and depth limited versions in 10 repeats. The result shows that as d increases, only deeper models can find the optimum solution. Therefore, as the depth of overlap increases, the depth of the neural network required to capture the problem structure efficiently also increases, as expected, confirming that these prob-



(a) Evaluating the performance of all MBOAs on C=OV, N=256 at different depth of overlap. Results show the number of function evaluations required to find a global optimum

(b) The best solution found by DO when using an autoencoder model limited to a maximum number of layers. DO requires a deep autoencoder model to capture the problem structure

Fig. 14 The effect of depth of overlap on the performance of an algorithm. LTGA and P3 are sensitive to the change in the depth of overlap (as the size of variation required to a find higher-order solution containing overlap increases). The depth of the model used by DO is also sensitive to the depth of overlap and consequently a shallow model is not capable of representing the problem structure to find a global optimum

lems have deep structure that must be learned and exploited to solve them, and that this is what the DO method is doing.

7.4 Performance Evaluation Summary

The MBOA distinctions we conclude from these experiments are:

1. Model-Informed Variation enables local search to be conducted in the new neighbourhood defined by the model. This enables DO to use the model in a way that the other methods (i.e. Model-Informed Crossover and Model-Informed Generation) cannot (see Fig. 11 and Fig. 12).
2. Overlap distinguishes the induction capability between tree and graph structured models. LTGA and P3 fail to learn and exploit overlapping variation operators and thus require exponential time complexity as the size of overlap increases. hBOA, DSMGA-II and DO are successful here (see Fig. 13).
3. Pairwise independent variation distinguishes the induction capability of DO and P3 from other MBOAs. LTGA, hBOA and DSMGA-II scale exponentially as the number of operators increases (see Fig. 10(c)).
4. A deep representation, and the ability to search in deep representations, are required when generating hierarchical combinations of overlapping operators (see Fig. 13) or mappings containing pairwise independent functions (see Fig. 10(c)) — distinguishing the capability of deep from shallow representations.

Prior explorations of these algorithms have shown cases where one is better than the other but have not previously shown a problem class that a subset of the algorithms can solve easily that the others cannot (here shown rigorously in the sense of

a polynomial vs exponential distinction in the scaling of their time complexities). By varying the complexity of the compression mapping and environment mapping we demonstrate how differences in the capability of model-induction or model-informed search of these algorithms can limit their performance.

Here we have identified problem challenges that differentiate the performance of MBOAs from DO, and are summarised in Table 15. Further research is required to understand if MKP instances contain the problem challenges identified here and if these challenges are responsible for the performance differences between DO and MBOAs in MKP. Similarly, we look forward to experimenting with DO on other binary combinatorial problems (e.g. 3-SAT), and developing extensions to order-based problems (e.g. flow-shop scheduling). Both of these will present further opportunities to understand the nature of the challenge these types of problems exhibit, and whether these map well onto the challenges identified here that give DO an advantage.

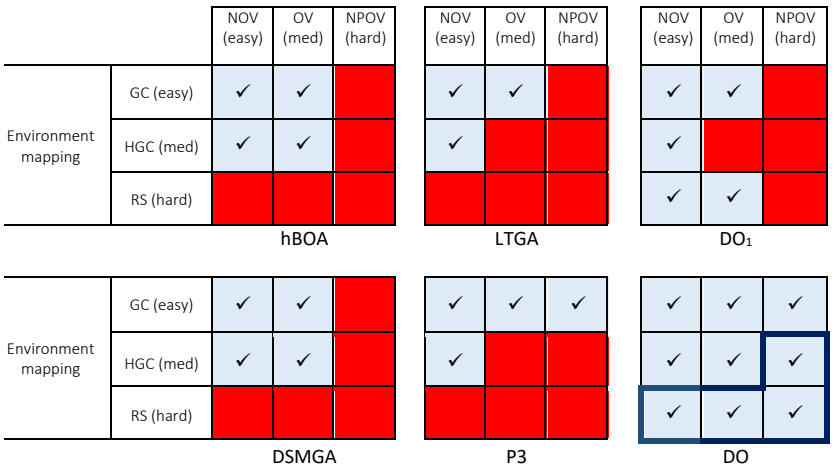


Fig. 15 The success or failure of each method to find a global optimum in polynomial time. DO is the only algorithm to find a global optimum in all problems. When either the environment mapping or the compression mapping is of the most difficult kind, none of the hBOA, DSMGA or P3 methods can find the global optimum in polynomial time and the P3 can only succeed for the difficult compression mapping when the environment mapping is easy. This leaves a set of problems (bold border) where DO is the only method to succeed. DO has different capabilities from DO_1 that is limited to a shallow model (a single layer) indicating that DO is succeeding because it is discovering and exploiting useful deep structure in these problems that the other algorithms cannot (see also Fig. 14)

8 Discussion

Our findings (particularly those in Sections 5, 6 and 7.4) merit some further brief reflections. First, we consider how varying the depth of the autoencoder(s) used in our study informs us about DO and other MBOA models (or variation schemes) concerning what is necessary and sufficient for learning and exploitation of complex

structure. We also consider what is not yet known. In a second subsection, we discuss how the models studied in this paper are (or may be) connected to biological evolution.

8.1 Autoencoder Model

The number of layers used by DO varied between experiments due to the complexity of a problem instance and the size of the problem instance. In the simplest case of non-overlapping dependencies (C=NOV), a single hidden layer is required to learn the problem structure. However, for large problem sizes it was observed that DO performed up to 2 transitions for non-hierarchical configurations and up to 3 transitions for hierarchical configurations. For the compression mapping containing overlapping dependencies (C=OV) a similar behavior was observed with DO performing up to 3 transitions for the non-hierarchical configurations. For the hierarchical configuration, the number of layers used by DO closely followed the number of layers in the problem. However, as observed in the non-hierarchical configurations, as the problem size increased the number of layers used by DO increased beyond the number of layers in a problem instance. It is not clear whether the increased number of hidden layers was necessary for large problem instances containing shallow problem structure. An alternative hypothesis is that the effect of updating the model with updated solutions, and consequently solutions containing a stronger signal for good combinations of variables, is what enables the induction of a representation that is more suitable for MIV. Results for DO_1 (the model limited to a single hidden layer) supports this hypothesis, however, further investigation is required as DO_1 also failed on some large problem instances.

Finally, for the C=nDOV complexity, DO often required at least 2 hidden layers to find the globally optimal for the smallest problems instance. However DO was not able to find solutions for large problem instances. For the large non-hierarchical configurations, we observed up to 5 transitions occurring. For larger instances that failed, we observed that the population of solutions was converging onto a sub-optimal solution. We hypothesise that the phenomena of catastrophic forgetting [30] is a contributing factor. Specifically, information learned at lower-levels that were found useful in early populations can be lost if a later population does not contain this signal (or this signal becomes weaker). This was observed particularly in the E=RS configuration.

Finally, our investigation primarily focused on separating the performance between the start-of-the-art MBOAs with respect to model induction and model informed search capabilities. In particular, we have focused on how a deep neural network can expand the class of MBOAs and consequently focused on what DO can do that other MBOAs cannot. To that end, we have not investigated the reverse. Specifically, what shallow models, or what other MBOAs, can do that DO cannot. It remains for future research to understand if there are other problem types that DO cannot solve (with polynomial scaling) but other MBOAs can. Understanding this would be valuable to help further the development of all MBOAs.

8.2 Connections with Biological evolution and processes of natural induction

These results show that multi-scale evolutionary processes can solve problems that single-scale evolutionary processes cannot. This suggests that in biological evolution, evolutionary processes that operate at multiple scales of biological organisation have adaptive capabilities that single-scale evolution does not. Although the methods employed in this paper use machine learning techniques that are unbiological (as is appropriate for practical optimisation motives), it is readily apparent that biological evolution does rescale the evolutionary process at successive levels of biological organisation. The basis of the relevant machine learning methods, including back-propagation, is the optimisation of an objective function through local improvement in model parameters. Accordingly, it makes sense that evolutionary processes effect comparable results if applied to similar model parameters. In evolutionary models, it is also clear that if there is heritable variation in the connections of a suitable network structure, the effect of selection on these connections is the same as that of basic connectionist learning [51]. This 'natural induction' has been shown in gene-regulation networks, social networks [11, 53] and ecological networks [37]. This might explain how biological evolution could achieve the model induction as well as the local search process using the induced model.

With these observations in mind, it is not implausible that biological evolution could be implementing something like Deep Optimisation, or deep natural induction, and the evolutionary transitions in individuality are the observable result [55]. If true, the notion of biological evolution as a simple local search process would be over simplistic. Whilst it might reasonably be described as a process of gradual incremental improvement at any one level of biological organisation, this description might miss the big picture. Contrary to its common characterisation as a plodding, unintelligent process, the type of problem-solving that biological evolution can perform, when taking account of its multi-scale nature, is potentially very different — more akin to a deep associative learning machine than local incremental improvement.

In DO, a similar kind of rescaling is exploited — enabling search that changes from searching combinations of primitive variables, to searching combinations of variables in a higher-level recoding of the solution space, and so on through multiple levels. The evolutionary model used is a simple hill-climber, but as higher-level representations are learned the hill-climbing process is repeatedly rescaled to operate in successively higher-level representations. The transition process is based on a deep learning neural network, specifically a deep auto-encoder. By training a simple encoder (with a single hidden layer) to compress a distribution of locally optimal solutions, it recognises the effective reduction in the degrees of freedom created by selection. The hill-climbing process then transitions to operate in the compressed representation, encoded in the latent variables of the hidden layer. Variation in this new space can make 'leaps' in the original solution space — coordinated changes of multiple variables, informed by the results of past search. Hill-climbing in this compressed space can still get stuck, but these intelligent leaps mean that the 'local' optima it finds are less limited than those found by hill-climbing in the original space. Additional transitions are enabled by adding further layers, trained on the local optima that are discovered using the layer below.

9 Conclusion

In this paper we have investigated the optimisation capabilities of an algorithm inspired by the Evolutionary Transitions in Individuality. In these transitions, the natural evolutionary process is repeatedly rescaled through successive levels of biological organisation. Each transition creates new higher-level evolutionary units that combine multiple units from the level below. We call the algorithm Deep Optimisation to recognise both its use of deep learning methods and the multi-level rescaling of biological evolutionary processes.

We started by testing DO in an applied problem. DO showed impressive performance when compared to other MBOA methods on MKP instances. This is all the more impressive bearing in mind that the other methods utilise a problem-specific repair operator to handle infeasible solutions and DO does not. This demonstrates that DO learns the structure of the problem well enough to enable exploration of feasible solutions directly. To go beyond this empirical result, we wanted to verify that DO was really able to find and exploit deep structure implicit in a problem space. The MKP instances on their own cannot confirm this because they have a random structure and, in addition to the use of a deep encoder, there are multiple other differences between DO and the other MBOA methods.

To investigate this we explored the types of problem structure that differentiate the performance between MBOA methods using synthetic problems. In these constructions we controlled several different problem characteristics, including the depth of the problem structure. We looked for characteristics that one algorithm could solve and another algorithm could not in the formal sense of a polynomial vs exponential time complexity, respectively. DO was the only method that solved all problem types (i.e. in polynomial time) where none of the other methods achieved this. This exploration identified that overlapping variation, non-pairwise overlapping variation and local search in the reorganised neighbourhood are distinct problem characteristics that DO can solve that other MBOAs cannot. The results showing that DO outperforms DO_1 (i.e., DO limited to a single hidden layer) verify that it is the deep representation, not other characteristics of DO on their own, that enables these results. That is, deep learning is needed, and DO learns deep problem structure and exploits it properly. None of the other SOTA methods can do this successfully.

In sum, our findings suggest that the use of deep learning principles, that have enjoyed success in many other domains, also have significant previously-untapped potential in optimisation problems. Thus far we have employed only fairly basic deep learning methods, but the DO approach opens up optimisation problems as another area where the vast and rapidly accelerating knowledge in deep learning techniques might be applied to significant effect.

Acknowledgements We acknowledge financial support from the EPSRC Centre for Doctoral Training in Next Generation Computational Modelling grant EP/L015382/1. JK acknowledges time granted to him by Invenia for pursuing personal research goals. Further, we would like to acknowledge Chrisantha Fernando, David Iclanzan, David Prosser, Frederick Nash, Rob Mills, Jakob Philbrick for enriching the ideas for DO.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Aickelin U, Burke EK, Li J (2007) An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. Journal of the Operational Research Society 58(12):1574–1585
2. Bello I, Pham H, Le QV, Norouzi M, Bengio S (2016) Neural combinatorial optimization with reinforcement learning. CoRR abs/1611.09940, URL http://arxiv.org/abs/1611.09940, 1611.09940
3. Bosman PA, Thierens D (1999) Linkage information processing in distribution estimation algorithms, vol 1999. Utrecht University: Information and Computing Sciences
4. Boyan J, Moore AW (2000) Learning evaluation functions to improve optimization by local search. Journal of Machine Learning Research 1(Nov):77–112
5. Caldwell J, Knowles J, Thies C, Kubacki F, Watson R (2021) Deep optimisation: Multi-scale evolution by inducing and searching in deep representations. In: Castillo PA, Jiménez Laredo JL (eds) Applications of Evolutionary Computation, Springer International Publishing, Cham, pp 506–521
6. Caldwell JR, Watson RA, Thies C, Knowles JD (2018) Deep optimisation: Solving combinatorial optimisation problems using deep neural networks. CoRR abs/1811.00784, URL http://arxiv.org/abs/1811.00784, 1811.00784
7. Ceberio J, Irurozki E, Mendiburu A, Lozano JA (2013) A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. IEEE Transactions on Evolutionary Computation 18(2):286–300
8. Chen PL, Peng CJ, Lu CY, Yu TL (2017) Two-edge graphical linkage model for DSMGA-II. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 745–752
9. Chu PC, Beasley JE (1998) A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics 4(1):63–86
10. Churchill AW, Sigtia S, Fernando C (2014) A denoising autoencoder that guides stochastic search. CoRR abs/1404.1614, URL http://arxiv.org/abs/1404.1614, 1404.1614
11. Davies AP, Watson RA, Mills R, Buckley CL, Noble J (2011) "if you can't be with the one you love, love the one you're with": How individual habituation of agent interactions improves global utility. Artificial Life 17(3):167–181
12. De Boer PT, Kroese DP, Mannor S, Rubinstein RY (2005) A tutorial on the cross-entropy method. Annals of Operations Research 134(1):19–67
13. Goldberg DE (2006) Genetic algorithms. Pearson Education India
14. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Machine Learning 3(2):95–99
15. Goldman BW, Punch WF (2014) Parameter-less population pyramid. In: Proceedings of the 2014 Annual Conference on Genetic and Evolution-

ary Computation, Association for Computing Machinery, New York, NY, USA, GECCO 14, pp 785–792, DOI 10.1145/2576768.2598350, URL https://doi.org/10.1145/2576768.2598350
16. Goldman BW, Punch WF (2015) Fast and efficient black box optimization using the parameter-less population pyramid. Evolutionary Computation 23(3):451–479
17. Hansen N (2006) The CMA Evolution Strategy: A Comparing Review. In: Lozano J, Larraaga P, Inza I, Bengoetxea E (eds) Towards a New Evolutionary Computation, Studies in Fuzziness and Soft Computing, vol 192, Springer, pp 75–102
18. Hansen P, Mladenović N, Pérez JAM (2010) Variable neighbourhood search: methods and applications. Annals of Operations Research 175(1):367–407
19. Hernández-Lobato JM, Gelbart M, Hoffman M, Adams R, Ghahramani Z (2015) Predictive entropy search for Bayesian optimization with unknown constraints. In: International Conference on Machine Learning, PMLR, pp 1699–1707
20. Hopfield JJ, Tank DW (1985) Neural computation of decisions in optimization problems. Biological Cybernetics 52(3):141–152
21. Horn J, Goldberg DE, Deb K (1994) Long path problems. In: International Conference on Parallel Problem Solving from Nature, Springer, pp 149–158
22. Hsu SH, Yu TL (2015) Optimization by pairwise linkage detection, incremental linkage set, and restricted/back mixing: DSMGA-II. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp 519–526
23. Iclanzan D, Dumitrescu D (2007) Overcoming hierarchical difficulty by hill-climbing the building block structure. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp 1256–1263
24. Khalil E, Dai H, Zhang Y, Dilkina B, Song L (2017) Learning combinatorial optimization algorithms over graphs. In: Advances in Neural Information Processing Systems, pp 6348–6358
25. Lombardi M, Milano M, Bartolini A (2017) Empirical decision model learning. Artificial Intelligence 244:343–367
26. Martins JP, Delbem AC (2016) Pairwise independence and its impact on estimation of distribution algorithms. Swarm and Evolutionary Computation 27:80–96
27. Martins JP, Neto CB, Crocomo MK, Vittori K, Delbem AC (2013) A comparison of linkage-learning-based genetic algorithms in multidimensional knapsack problems. In: 2013 IEEE Congress on Evolutionary Computation, IEEE, pp 502–509
28. Martins JP, Fonseca CM, Delbem AC (2014) On the performance of linkage-tree genetic algorithms for the multidimensional knapsack problem. Neurocomputing 146:17–29
29. Mazyavkina N, Sviridov S, Ivanov S, Burnaev E (2020) Reinforcement learning for combinatorial optimization: A survey. CoRR abs/2003.03600, URL https://arxiv.org/abs/2003.03600, 2003.03600
30. McCloskey M, Cohen NJ (1989) Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol 24, Elsevier, pp 109–165

31. Mills R, Watson RA (2011) Multi-scale search, modular variation, and adaptive neighbourhoods. Author's Original
32. Ollivier Y, Arnold L, Auger A, Hansen N (2017) Information-geometric optimization algorithms: A unifying picture via invariance principles. Journal of Machine Learning Research 18(18):1–65
33. Pelikan M, Goldberg DE (2003) Hierarchical BOA solves ising spin glasses and MAXSAT. In: Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation: PartII, Springer-Verlag, Berlin, Heidelberg, GECCO'03, pp 1271–1282
34. Pelikan M, Goldberg DE (2006) Hierarchical Bayesian optimization algorithm. In: Scalable optimization via probabilistic modeling, Springer, pp 63–90
35. Pelikan M, Goldberg DE, Cantú-Paz E, et al. (1999) BOA: The bayesian optimization algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, vol 1, pp 525–532
36. Pelikan M, Goldberg DE, Tsutsui S (2003) Hierarchical Bayesian optimization algorithm: toward a new generation of evolutionary algorithms. In: SICE 2003 Annual Conference (IEEE Cat. No. 03TH8734), IEEE, vol 3, pp 2738–2743
37. Power DA, Watson RA, Szathmáry E, Mills R, Powers ST, Doncaster CP, Czapp B (2015) What can ecosystems learn? Expanding evolutionary ecology with learning theory. Biology Direct 10(1):1–24
38. Probst M (2015) Denoising autoencoders for fast combinatorial black box optimization. CoRR abs/1503.01954, URL http://arxiv.org/abs/1503.01954, 1503.01954
39. Rodríguez Rueda D, Cotta C, Fernández-Leiva AJ (2021) Metaheuristics for the template design problem: encoding, symmetry and hybridisation. Journal of Intelligent Manufacturing 32(2):559–578
40. Santana R (2017) Gray-box optimization and factorized distribution algorithms: where two worlds collide. CoRR abs/1707.03093, URL http://arxiv.org/abs/1707.03093, 1707.03093
41. Santana R, Larrañaga P, Lozano JA (2008) Protein folding in simplified models with estimation of distribution algorithms. IEEE Transactions on Evolutionary Computation 12(4):418–438
42. Smith JM, Szathmáry E (1997) The major transitions in evolution. Oxford University Press
43. Snoek J, Rippel O, Swersky K, Kiros R, Satish N, Sundaram N, Patwary M, Prabhat M, Adams R (2015) Scalable bayesian optimization using deep neural networks. In: International Conference on Machine Learning, PMLR, pp 2171–2180
44. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2):99–127
45. Terashima-Marín H, Ross P, Farías-Zárate C, López-Camacho E, Valenzuela-Rendón M (2010) Generalized hyper-heuristics for solving 2d regular and irregular packing problems. Annals of Operations Research 179(1):369–392
46. Thierens D, Bosman PA (2013) Hierarchical problem solving with the linkage tree genetic algorithm. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, pp 877–884

47. Vo-Thanh N, Jans R, Schoen ED, Goos P (2018) Symmetry breaking in mixed integer linear programming formulations for blocking two-level orthogonal experimental designs. Computers & Operations Research 97:96–110
48. Volpato R, Song G (2019) Active learning to optimise time-expensive algorithm selection. CoRR abs/1909.03261, URL http://arxiv.org/abs/1909.03261, 1909.03261
49. Vu KK, D'Ambrosio C, Hamadi Y, Liberti L (2017) Surrogate-based methods for black-box optimization. International Transactions in Operational Research 24(3):393–424
50. Watson RA (2005) On the unit of selection in sexual populations. In: Capcarrère MS, Freitas AA, Bentley PJ, Johnson CG, Timmis J (eds) Advances in Artificial Life, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 895–905
51. Watson RA, Szathmáry E (2016) How can evolution learn? Trends in Ecology & Evolution 31(2):147–157
52. Watson RA, Hornby GS, Pollack JB (1998) Modeling building-block interdependency. In: International Conference on Parallel Problem Solving from Nature, Springer, pp 97–106
53. Watson RA, Buckley CL, Mills R (2011) Optimization in self-modeling complex adaptive systems. Complexity 16(5):17–26
54. Watson RA, Mills R, Buckley CL (2011) Transformations in the scale of behavior and the global optimization of constraints in adaptive networks. Adaptive Behavior 19(4):227–249
55. Watson RA, Levin M, Buckley CL (2021) Design for an individual: Connectionist approaches to the evolutionary transitions in individuality. Frontiers in Ecology and Evolution, section Social Evolution
56. West SA, Fisher RM, Gardner A, Kiers ET (2015) Major evolutionary transitions in individuality. Proceedings of the National Academy of Sciences 112(33):10112–10119
57. Yi S, Wierstra D, Schaul T, Schmidhuber J (2009) Stochastic search using the natural gradient. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp 1161–1168
58. Zhang W, Dietterich TG (2000) Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. Journal of Artificial Intelligence Reseach 1:1–38
59. Zlochin M, Dorigo M (2002) Model-based search for combinatorial optimization: A comparative study. In: International Conference on Parallel Problem Solving from Nature, Springer, pp 651–661