UNIVERSITY OF SOUTHAMPTON

Deep Optimisation: Learning and Searching in Deep Representations of Combinatorial Optimisation Problems

by

Jamie R. Caldwell

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

December 2020

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Jamie R. Caldwell

Finding solutions to challenging optimisation problems in a feasible time requires intelligent search. Often, this intelligence is provided by humans, however the design of complex solutions has been achieved by non-human intervention. Namely, solutions found by natural evolution. With recent connections between machine learning and evolutionary process, and specifically the evolutionary transition in individuality, this thesis develops a novel Model-Building Optimisation Algorithm (MBOA) called Deep Optimisation (DO) that induces a multi-level representation of a combinatorial optimisation problems using a deep neural network. This multi-level representation allows for the process of variation and selection to perform at multiple-levels of organisation.

DO is within the framework of Model-building optimisation algorithms (MBOAs) that are an extension of evolutionary algorithms. MBOAs use machine learning methods to adaptively reorganise the neighbourhood of an evolutionary search space. This can allow for efficient navigation through a fitness landscape; finding solutions that would otherwise be pathologically difficult to find. Relative to current deep learning methods, the models used by the state-of-the-art (SOTA) MBOAs are of lower sophistication and capacity. Deep Optimisation (DO) is a novel MBOA that uses an autoencoder model to recursively transform the neighbourhood of a solution to high-orders of organisation. It then searches within these representations using Model-Informed Variation (MIV) to improve a candidate solution. Due to the combination of a deep representation and MIV, DO is able to find solutions that the SOTA-MBOAs cannot.

DO emulates the transition in individuality by using a deep autoencoder model to transform the representation of a solution by capturing associate relationships between lower-level units that contribute to the quality of a solution. The transformed representation enables variation and selection to act on individual hidden nodes that transform back to the solution level as a higher-order variation. The transformation is repeatedly applied, constructing a deep representation of the solution, and enabling search at multiple scales of organisation. Thus, in effect, DO performs a recursive reorganisation of the neighbourhood of a solution by inducing representations that capture relationships that contribute to the quality of a solution.

This thesis successfully identifies problem characteristics that distinguish the performance between the SOTA-MBOAs and also DO with regards to both the model capacity and model exploitation. Experiments show that DO is capable of finding solutions to problems in polynomial time that other SOTA-MBOAs require exponential time. The identified challenges are overlapping variation operators, pairwise independent variation operators and cyclic paths. Further, it is found that there exists deep problem structure that can be overcome in polynomial time using a deep model but take exponential time when using a shallow model.

Finally, DO is applied to different optimisation problem domains to demonstrate its potential for exploiting problem structure in problems with unknown structure. This thesis provides a connection between deep learning models and MBOAs, showing state-of-the-art results can be achieved by utilising the tools available in deep learning. This suggests numerous avenues for further investigation, transferring state-of-the-art deep learning methods into the domain of MBOAs

Contents

List of Figures

List of Tables

Listings

Nomenclature

w	The weight vector

Acknowledgements

We acknowledge financial support from the EPSRC Centre for Doctoral Training in Next Generation Computational Modelling grant EP/L015382/1. This work is partially supported by Templeton project 60501.

To …

Chapter 1

Introduction

Idea of incorporating more mechanisms from biology into evol computing ETI not evolvability What class of algorithms, within the state of the art MBOAs

Evolutionary computing is the study of computational algorithms that emulate evolutionary processes to find good solutions to an optimisation problem. Specifically, an organism (solution) is adapted via variation and selection (search operators) to maximise (optimise) the survival in an environment (problem). The dynamical process of evolution provides the mechanism of adaptation to a solution to the environment - finding a solution of high quality. Evolution by natural selection has proven to provide extraordinary adaptations to an organism in a variety of challenging environments. It is with this observation that evolutionary computing has been extensively studied for solving optimisation problems De Jong (2006); Eiben and Smith (2015)

Understanding the mechanisms that provide adaptation via evolutionary process remains an extensive area of research both in evolutionary biology Laland et al. (2015) and evolutionary computation Eiben and Smith (2015). It is hypothesised that in Genetic Algorithms (GA), evolvability - the 'ability of random variation to sometimes produce and improvement' Wagner and Altenberg (1996) - is evolved by conserving low-order adapted variable combinations whilst maintain variability between them. This is formally known as the Building-Block Hypothesis (BBH) Goldberg (1989); Holland et al. (1). It is hypothesised that the exchange or accumulation of building-blocks is likely to improve the fitness of a solution. In doing so, the dimensionality of the search space has been adapted from variation between individual units to variation between combinations of adapted units. Or, rather, variability is transformed to a higher level evolutionary unit.

The BBH provides a thought-provoking mechanism for evolving variability to maintain adaptive variation throughout the evolution of the population. Previous experience of variable combinations that contributed to the quality of a solution early in the search are used to guide future variation, rather than performing random exploration. However, in GAs, the BBH has not been proven beyond the first generation and is further limited to low-order (short) building-blocks, where the likelihood of disruption is small O'Reilly and Oppacher (1995); Beyer (1997). Further, for a hierarchical construction of building-blocks, where lower-level building-blocks combine to construct a higher-level building-block, it requires variables within a building-block to be located physically close at the solution representation Thierens and Goldberg; Thierens (1995, 1999); Watson et al. (1998).

Significant attention has been given to improving evolutionary computation by using machine learning to explicitly capture and exploit the building-block information and overcome the challenge of positional bias during recombination Hauschild and Pelikan (2011). As we explore in this thesis, the interpretation of what the model learns and represents, how the model is used, and how this connects with biological evolutionary process varies between algorithms. We classify these algorithms as Model-Building Optimisation Algorithms (MBOAs). Within this class, algorithms can be broadly separated into Estimation of Distribution Algorithms (EDAs), that use a probabilistic model to replace crossover and mutation operators in a GA and share little resembelence to evolutionary processes Pelikan and Goldberg (2006); Thierens (2010); Goldman and Punch (2014); Hsu and Yu (2015), and Multi-Scale Search Algorithms, that use machine learning models to bias the variability of an individual solution Iclanzan and Dumitrescu (2007); Mills et al. (2014); Cox (2015); Watson et al. (2011) and are connected to the evolution of developmental processes Watson et al. (2014). In each case, a machine learning model is used to capture relationships that contribute to a solutions quality and used to bias future search.

In this thesis we explore how the theory of Evolutionary Transitions in Individuality (ETI) West et al. (2015) can be emulated in evolutionary computing. The theory of ETI describes a mechanism for the evolution of evolvability through a multi-scale process where individual evolutionary units at one level of organisation form associations that result in a new evolutionary unit at a higher level of organisation Smith and Szathmáry (1997); Watson and Szathmáry (2016). These emergent higher-order entities become themselves subject to natural selection, allowing for combinations of units from the previous layer to be combined and selected together and thus enabling evolution to perform at a new level of representation. Further, this process is recursively applied, producing successive hierarchical transitions in individuality Watson and Szathmáry (2016). Therefore, unlike the methods that use machine learning methods to replace the recombination operators used in GAs Hauschild and Pelikan (2011), we explore how a deep model can be used to emulate the developmental process from genotype to phenotype, which in turn biases the variability of an individual solution Watson et al. (2011); Mills et al. (2014); Cox (2015). Deep Neural networks provide a model space to explicitly capture and represent a multi-level representation of a solution with each layer inducing higher-order representations from the layer below. Further, recent success in deep learning has provided a scalable method for constructing deep representations of a data set Hinton and Salakhutdinov (2006). In doing so, they have produced state-of-the-art (SOTA) results for many tasks Krizhevsky et al. (2012); Hinton et al. (2012a); Cho et al. (2014); Silver et al. (2016), but not including optimisation. Inspired by the connections between developmental processes and associative learning Watson et al. (2014); Watson and Szathmáry (2016), in this thesis, we connect deep neural network models in the process of evolutionary computing to emulate ETI. We call this approach Deep Optimisation (DO). A significant difference between DO and other MBOAs is the ability to represent a solution at multiple levels of organisation and to perform variation and selection at this level of organisation. In DO, solution variables are encoded into a compressed hidden space which represents a relationship between the units in the layers below that contribute to the solutions quality identified by previous experience. Variation and selection is performed at this induced representation by performing a local variation at the hidden layer and decoding back into the solution space, producing large but organised variation. 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. A deep model is constructed by the recursive process of inducing a higher-level representation of solutions found by variation and selection performed at the representation level below, in the analogue of successive hierarchical transitions in individuality Watson and Szathmáry (2016).

The idea of depth in a model is unique to Deep Optimisation: all state-of-the-art MBOAs use the model to capture relationships between the original solution variables. This leads to the research questions:

How can a deep neural network
improve evolutionary optimisation?

As we explore in this thesis, current SOTA MBOAs differ significantly in the model capacity, method for exploiting information from the model, and their biological interpretation. In this thesis we investigate the types of problem characteristics that DO can overcome that other MBOAs cannot. Further, during our investigation, we construct a theoretical problem that contains overlapping dependencies between building-blocks that, for the first time, that categorically differentiates the performance between existing SOTA MBOAs. Further, we find that the characteristics of cyclic paths categorically differentiate the performance betwen methods used for exploiting model information and pairwise independent functions categorically differentiate the performance between model capacities. In conclusion, Deep Optimisation is the only algorithm that does not exhibit exponential running time on any of these problem characteristics. We then proceed to explore how DO works and show that higher order representations of a solution provide an adaptively reorganised neighborhood of a solution, presenting a smoother and simpler landscape for variation and selection to navigate, finding solutions with high-quality that were otherwise pathologically difficult to find. Further, we show using a single hidden layer, and not a deep layered network, fails to efficiently overcome the problem characteristics of overlap or pairwise independence. Finally, we apply DO to mainstream optimisation domains and discuss DOs suitability and further directions.

The contributions of this thesis are:

1. Deep Optimisation is able to find solutions in polynomial time that for state-of-the art MBOAs takes exponential time with respect to functions evaluations.
2. Using the model to bias the variability of an individual solution is able to find solutions in polynomial time that when using the model to bias the variability of a population takes exponential time with respect to functions evaluations
3. Variation and selection performed at a deep representation of a solution can overcome problem characteristics in polynomial time that for variation and selection performed at a shallow representation takes exponential time.
    (a) The problem challenges of hierarchical overlapping variation is solvable in polynomial time using a deep neural network but takes a shallow neural network exponential time.
    (b) The problem challenge of higher-order dependencies requires a multiple level representation in a neural network model to represent a compressed representation of the search space.
    (c) The problem challenge of repeated optimisation requires information of previous neighbourhood representations to be maintained and separated.

1.1 Thesis Overview

The thesis is organised as follows:

Chapter 2: The foundations of this thesis are discussed and we review the relevant literature regarding? the development of evolutionary computing, with specific attention to the use of machine learning methods.

Chapter 3: The functionality of MSSA and EDA are compared using the algorithms rHN-G and BOA. We show that rHN-G can overcome problem challenges that BOA cannot even though the model used in BOA is of higher sophistication than rHN-G. We then develop the rHN-G algorithm to use an autoencoder model, rA-G, and demonstrate a comparable performance with rHN-G. We show that rA-G can represent problem structure that rHN-G cannot and that synthetic hierarchical problems, previously used as a benchmark in the literature for MBOAs, can be represented by a shallow model.

Chapter 4: Introduction of the Deep Optimisation (DO) Algorithm. We explore the relevant literature that is applicable to the understanding of inducing and representing higher-order representations of a dataset. The DO algorithm is then described in detail, providing information about alternative design decision that are later explored in this thesis. Finally, we compare the functionality of DO with all state-of-the-art methods used as comparative algorithms in this thesis. We show that significant differences exists between their functionalities and develop hypothesis for problem characteristics that categorical separate the performance between algorithms into can and cannot solve.

Chapter 5: We first review the relevant literature for constructing and using synthetic benchmark optimisation problems. We develop and detail the theoretical problem construction that contains problem characteristics identified in Chapter 4 into a single problem. We explore the interaction between these characteristics to understand the challenges the problem presents to an optimisation algorithm.

Chapter 6: We perform a comprehensive scalability study using the theoretical problem. The result show a categorical differentiation between the performance of the existing state-of-the-art algorithms for the first time. Further we show a categorical differentiation between the state-of-the-art MBOAs and DO. DO is the only algorithm capable of efficiently overcoming all problem challenges. (possible merge with Chapter 5)

Chapter 7: We perform an in-depth analysis on the performance of DO, with specific attention to how DO is exploring the solution landscape, and how the induction of higher-order representations provides a simpler landscape for variation and selection to explore. (not sure how relevant this chapter is)

Chapter 8: In the final chapter, we apply DO to different optimisation domains and show promising results and future directions to explore. We further discuss other areas of application for DO and how DO can provide unique functionality to different optimisation domains.

Chapter 2

Foundations

There exist many algorithms that are suitable methods for finding a solution to an optimisation problem. This thesis is interested in search methods that are explicitly aligned with the bias of the building-block hypothesis. Specifically, algorithms that automatically reduce the dimensionality of the search space by exploiting relationships between variables that contribute to a solutions quality. To this regard we find methods that implicitly or explicitly exploit this bias of search space exploration. This thesis is particularly focused on methods that use machine learning methods to explicitly capture and exploit these regularities. We find that no state-of-the-art performance has been achieved using neural network methods.

In this chapter, the foundations that lead to the development of machine learning models to exploit the build-block hypothesis and the state-of-the-art methods are reviewed. This provides the concepts that are built-upon in this thesis.

This chapter provides the main literature review, from which the main contributions of this thesis are developed from. For readability, Chapters 5 provides additional review of the synthetic problems used in the evolutionary computation community for distinguishing the performance of MBOAs. Chapter 4 provides additional review of deep learning models that are suitable for use in Deep Optimisation. Finally, in Chapter 7, additional literature is reviewed regarding applied problems.

2.1 Concepts

2.1.1 Combinatorial Optimisation Problems

Combinatorial optimisation (CO) is the task of searching for a solution from a finite collection of possible candidate solutions that maximises (or minimises) the objective function. The objective function (also referred to as the fitness function) provides a measure for a solutions quality. CO problems contain the characteristic of a discrete variables and thus are non-differentialable problems. This makes them particularly challenging. For the challenging problems, we often rely on search methods to explore the solution space for a solution to the problem.

Many interesting CO problems fall into the class of NP-hard. A problem class refers to the description of a problem, the specific parameters of the problem such as the size of the problem are not specified. A problem instance is a problem described by the description of a problem class that has all parameters specified. The complexity of a problem class is defined by the Turing Machine computational model. Specifically, the complexity is a measure of the dependency between the time or space resources used to find a solution and the size of the problem instance. If the relationship is polynomial using a deterministic machine, the problem class is in complexity class P. if the relationships is polynomial using a non-deterministic machine, the problem class is in complexity class NP. A deterministic machine can be simulated using a non-deterministic machine and therefore a problem class in P is also in NP. However, the reverse has not been proven and remains one of computer sciences most famous unanswered questions, although most experts agree that $P \neq NP$. Interesting and practical problem classes are generally in NP but not in P, for instance SAT citeCook (1971); Garey and Johnson (1979), therefore using a deterministic machine the computational resource grow exponential with the size of the problems. Therefore, for heuristic methods are often used to search the solution space as they provide an efficient method for find solution (polynomial time complexity) although have no guarantee for finding the global optimal solution.

The complexity of a problem class is determined by the worst case performance on the problem class. Specifically, the worst case of problem instances in the problem class. However, they may exist problem instances with in a class that contains characteristics that can be solved in polynomial time, such as large real-world travelling salesman problems Applegate et al. (2006), or good solutions (that are regarded as sufficiently close to the optimal solution) can be found in polynomial time. Identifying these characteristics often requires years of dedicated research and domain expert knowledge to understand the characteristics of the problem such that methods can be developed to exploit this information.

In the case where problem structure is complex or unknown (such as the case in simulations) it is not possible to use algebraic model-based mathematical programming. Instead, black-box optimisation methods are required, where the term black-box refers to the problem structure of an optimisation problem and not the search algorithm. Therefore identifying exploitable problem structure requires an adaptive search search that responds to the characteristics of the search identified during optimisation.

The complexity of an algorithm is determined by measuring the proportional relationship between the change in computational effort to find a global optimal solution and the change in problem size. This provides a measure for the feasibility of an algorithm. An algorithm that shows exponential scaling, c^n, will inevitably become infeasible as the dimensional of the problem increases. In this case, we refer to the algorithm as failing. It is unable to overcome the problems challenges in the optimisation problem to efficiently find a solution. On the other hand, if an algorithm shows polynomial scaling, n^c, then the algorithm called successful (or efficient) as it is able to overcome the problem challenges to efficiently find a solution to the optimisation problem. We refer to algorithms only showing an efficiency difference if there is only a change to the the exponent and coefficient only (the base does not change).

In this thesis a solution referrers to an configuration to the optimisation problem. A globally optimal solution explicitly refers to the solution that has the highest quality out of the set of all possible candidate solutions. A locally optimal solution referrers to a solution that has a neighbourhood of solutions that all have lower quality. A candidate solution referrers to a solution that is being explored or utilised by an optimiser. A neighboring solution is a solution that given a variation operator is generatable from the current solution. A promising solution refers to a solution that shows greater than average quality relative to other other candidate solutions.

A fitness landscape provides a abstract visualisation method for representing the problem challenges that are induced by the problem structure. The landscape surface is defined by the solutions neighbourhood and the fitness differences between the neighbourhood solutions. Therefore one can visualise how exploiting information during search will effect the trajectory of a search (the path a solution takes during search). The characteristics of the landscape present challenges that an algorithm must overcome to navigate to a superior solution efficiently. A solutions neighbourhood is defined as the alternative solutions that are accessible from a current solution. The accessible solutions from a solution (its neighbours) are defined by the variation operator. This thesis uses the term variation operator to describe the change applied to a solution during search. The variation operator is relative to a candidate solution. For example, a single-point variation provides variants of a solution that are a single-point away. Of course, a solutions neighbourhood could be all solutions, as defined by K-point mutation, where k is the size of the problem N. However, the likelihood of applying the correct k-point mutation scales exponential with k

2.1.2 Black-Box Optimisation

Human-designed organisation requires domain-expert knowledge to decompose the problem at its natural joints. However, in this thesis, we are interested in algorithms that automate this decomposition. Natural evolution provides an inspiring example of an algorithm that exploits natural problem structure. Natural evolution is a dynamical process that cannot select for future variability that produces fitter individuals, yet natural evolution is able to produce a seemingly intelligent solution to the challenging environments. Multi-cellular organisms, found by natural evolution, show a remarkable organisational structure. Of course, these solutions may not be optimal for the environment. However, they are solutions of higher quality that have been efficiently found (considering the dimensionality of the original problem space). Effectively searching for a solution to an optimisation problem generally requires encoding some knowledge of the problem into the search process. Wolpert and MacReady put this succinctly with the No free lunch theorem Wolpert and Macready (1997). It states that the performance of a search process, averaged over all optimisation problems, will be equal to the performance of random search. This does not dictate an algorithm cannot perform better than random search. It simply states that a search process that has better than random performance on subset of problems will show a poor performance on problems outside this subset. The no free lunch theorem is not a limiting factor when designing optimisation methods. Rather, it is an important caveat to consider. Generally, we are interested in designing search methods that perform well on a specific class of problems. The performance of a search process is therefore assessed as the performance achieved on this distribution of problems, or as Wolpert describes how well aligned a search algorithm is with the distribution of problems Wolpert (2013). The alignment of a search algorithm is in reference to the ability of the search algorithm to exploit the problem structure.

Alignment of the search bias with the problem requires learning and understanding the nature of the problem subset and exploiting characteristics of the problem structure. For instance, if an optimisation problem can be modeled using a linear algebraic model, the global solution can be found in polynomial time **?**. Thus linear programming exploits the particular characteristic of linear relationships in the problem structure. For problems that cannot be represented by algebraic model-based mathematical programming, heuristic methods are used and developed to exploit particular problem structure. A successful example is the Lin-Kernighan heuristic used for the symmetric travelling salesman problem Lin and Kernighan (1973); Applegate et al. (2006). However, without domain specific knowledge, the ability to use algebraic models, or simply unknown problem structure (such as simulations) generally requires black-box optimisation methods.

Developing optimisation algorithms to find solutions in a feasible time (polynomial time complexity) requires learning and understanding the problem characteristics of the problem subset and then developing algorithms that align the search bias with the problem structure. This can be challenging and even impossible for complex optimisation problems. Specialised techniques that are capable of solving optimisation problem in a feasible time require expressing the problem using a particular form i.e., linear programming **?**, symmetric distance between cities (in the case of the travelling salesman problem)Lin and Kernighan (1973); Applegate et al. (2006). If an optimisation problem can be reduced to such form, their exist specialised and efficient solvers that can exploit particular characteristics that are present in algebraic model. However, for complex optimisation problems it can be impossible or infeasible to represent the optimisation problem accurately in such form. Further, the problem structure can simply be unknown as in the case of simulation based problems. In this case, the optimisation problem is said to be a black-box. In this case, developing algorithms that exploit the problem structure must infer this information from observations of the functions response (output) to changes in the solution (input) as illustrated in Figure 2.1. This thesis focuses on the optimisation methods that are suitable for black-box optimisation problems.

FIGURE 2.1: In a Black-Box Optimisation (BBO) problem, the relationships that compute the objective function (output response) for a given solution (input) are hidden from the optimiser. An intelligent optimiser must induce the black-box structure using the input and outputs only.

For constraint handling including violation of constraint, penalty term in to the objective function.

For real-world problems, developing optimisation algorithms to find solutions with high quality in a feasible time (polynomial time complexity) is the primary goal. Of course, it is preferred if the solution with highest quality is found. However, in practice this may not be possible, or even verifiable in polynomial time. For example, the solution for for benchmark problem instances in challenging optimisation problem is often refereed to as best known solution rather than global optimum as it is infeasible to verify to be the global optimum solution Burkard et al. (1997). Never-the-less efficient algorithms can still be useful for finding good quality solutions. In this thesis, we are interested in algorithms that exploit regular occurring relationships between variables that contribute to a solutions quality, and specifically combing and conserving these relationships to construct superior solutions.

2.1.3 Decomposable Problem Structure

In optimisation - the interactions is to increase fitness by satisfying the constriaints between the variables. The emergence of order in is related to the emergence of variable combinations that contribute to the fitness of a solutinon.

In this thesis, problem structure is used to refer to relationships between variables that contribute to a solutions quality. Often, it is not practical or even possible (in the case of black-box optimisation problems) to directly examine the problem structure . The question therefore arises. How can we exploit problem structure if we cant have access to it? Simons Simon (1969) explains that we can induce the natural decomposition of a function by observing its response to a stimulus. By stimulating the system, we can observe regularities in the response to help induce the structure of the function.

This structure is intuitive to an engineer. An engineer develops a global solution to a problem by decomposing the problem into smaller, easy to solve problems - a modular decomposition. Solution to these modules are then combined to form-higher solutions, an organisation of higher-order modules. The solutions to these higher-order modules are then recombined further, repeating the process to generate a global solution to the problem.

Efficient search for black-box optimisation problems requires biasing the search process based on few assumptions about the problem. In this thesis, the bias of the search is based on decomposable functions. Taking a complex high-dimensional function and reducing it to many smaller function that interact to form the original function is a principle adopted by engineer's. Generally, when an engineer is presented with a challenging optimisation problem they decompose the global problem it at its natural joints into smaller, easier to solve problems and then constructs a global solution from the solutions to these sub-problems. A process called problem decomposition. It is a reductionist technique that is a popular approach for optimisation Papadimitriou and Steiglitz (1998). It is assumed that if one can perform an effective reduction, one can expect to find good solutions to complex optimisation problems efficiently. The caveat being, of course, does the problem have a tractable decomposition that can be exploited efficiently. The decision and therefore knowledge required to make a successful decomposition of an optimisation problem is the following:

- Identifying sup-problems of the global optimisation problem - problem decomposition
- Identifying solutions to these sub-problems - partial solutions
- Combining partial solutions to construct a global solution - exploiting partial-solutions

For black-box optimisation methods one must infer the problem decomposition by observing the response of the function to variation in solutions to the problem. Simon Simon (1969) identifies that in naturally occurring functional systems, be that biological or human-designed (For example, multi-cellular organisms, software, engineered solutions and social organisation), share a nearly-decomposable structure. Specifically, the high-order organisation of a system (global function) is a resultant interaction between a nesting of many smaller low-order systems (sub-functions). The internal relationships of a sub-function (interactions between units within a sub-function) are stronger than the relationships with external sub-functions (interactions between units of different sub-functions). Therefore, the system can be decomposed into different scales of complexity and organisation, depending on the strength of interactions. The internal relationships are therefore prioritised in the system. Breaking this relationships, and therefore the sub-function, removes the external dependencies. An external dependency can only act on a viable partial solution. If there is no valid solution for the sub-problem, then with out 'a priori' information it is not possible to identify the external relationships of the partial solution.

The units within a sub-function are stronger than interactions with units outside the sub-function. Therefore, the system can be decomposed into different scales of complexity and organisation, depending on the strength of interactions between units. Simon Simon (1969) explains that we can induce the natural decomposition of functions by observing its reaction to stimulus. By stimulating the system, we can observe regularities that occur in the response that provide a signal to the type of decomposition of the complex system, or the features of a complex system (a combination of units that show a regular relationships when stimulated). Regularities that occur often can be inferred as contributions to the functionality of the complex system. For an optimisation problem, the objective function (system) is stimulated by make a change to a solution or comparing variant solutions (a change in the input) and observing the change to the objective value (the response of the system). In doing so, this provides a signal for the natural structure of the system that we can then use to infer the relationships between variables that contribute to variations in the solutions. It is assumed that a bottom-up construction that exploits sub-functions will produce a function of superior quality.

Connecting this with our thinking for deep optimisation. It is common to attribute the success of a deep neural network to its ability to extract salient features from a data-set, where each layer provides a higher-order abstraction of the features. For instance, in image recognition, the lower layers identify edges, the next layers combine edges to form regular shapes and the deepest layer uses the shapes to construct high-order objects. Indeed it's importance has formed a dedicated sub-field called representation learning that is further explored in chapter 4. The function of classification is a decomposition of nested simpler functions. Therefore deep neural networks appear a natural fit to explore the idea of decomposing an optimisation problems. However, the significant challenge for optimisation is the learning signal. This thesis uses inspiration form natural evolutionary process to provide the learn-able signal. Specifically, the stimulus and response is provided by variation and selection.

Of course, exploiting the natural structure may not align with the actual problem structure - the interactions between variables that contribute to the global solution. For instance, problems that contain deceptiveGoldberg (1987); Whitley (1991) or random structure Heckendorn et al. (1999). In this case, the relationships that appear due to the dynamics of evolutionary processes, namely natural selection, will not improve the search process. Therefore, exploiting natural problem structure will show and exponential scaling to find to global optimum. If the problem has optimal substructure - optimal solutions to sub-functions form the global solution - then we expect the algorithm to find the solution in polynomial time by exploiting the natural structure. It is expected that in generally optimisation problems, that problems are between these extremes. Therefore whilst exploiting natural problem structure may not provide a global optimal solution. It will proved a method for finding a natural solution in feasible time - the performance will be determined by the type of structure in a problem instance. For example, give an engineer a problem with a non-logical set of constraints, we would not expect them to easily find a solution. The bias of the engineering mindset makes them a superior performer for structured problems. Therefore we can expect that search process that exploits patterns and structure in an optimisation problem will have better than random performance on all types of problems that contain these patterns and structures. This thesis is interested in methods that automatically identify the problem structure for a problem instance and exploit this information to improve the search process.

If there is variation, then the unit must be variable. However, to opposite is not true. The measure of variation is only a surrogate for variability Wagner and Altenberg (1996). Therefore, modelling the variation in a distribution does not provide use with the variability in a dataset....

In this thesis, the type of problem structure explore is near-decomposable building-block structures. However, as discussed in Chapter, the devleopment of DO facilitates the exploitation of other types of important structure, such as backbones Prugel-Bennett (2007)

Watson and Knowles paper in here - multi-collectivisation single-objective optimisation problems. - something about forcing a decomposition of the problem and optimising this seperatly. Knowles et al. (2001)

Searching in

Mills et al. (2014) -

The idea of decomposing a global problem into simpler easy to solve problems is also the foundation of dynamic programming. Optimal sub-structure an non-overlapping problem structure can be solved using divide an conquer. If the problem contains overlapping sub-problems, then dynamic programming is applicable. (applicable to shortest path - bellman-ford algorithm).

Solving a problem by dividing it into smaller more manageable problems (in a bottom up process) assembling good solutions to small problems to find good solutions to large problems. divide and conquer, branch and bound

An optimisation problem that contains optimal sub-structure means that the global solution is combination of sub-problem optimal solutions.

Divide and conquer, and multi scale search: Solving a problem by dividing it into smaller more manageable problems (in a bottom up process) assembling good solutions to small problems to find good solutions to large problems.

Modular variation instead of singular

We assume that solutions to real-world problems, that are . Thus finding the correct representation of a solutoin that allows for modular exploration of the solution spcae is often used for optimising.

Superior solutions can be constructed from a set of promising solutions - regular occurring relationships between variables that contribute to a solutions quality is assumed to be an important relationships that is present in superior solutions.

2.2 Algorithms inspired by the processes of biological evolution.

A particular advantage of evolutionary computation in comparison to alternative optimisation methods is that it makes few assumptions about the problem structure. This makes the class of algorithms successful in finding solutions across a wide range of problem domains De Jong (2006). Therefore, by understanding and emulating the mechanisms that contribute to adaptation, we can further develop the success of evolutionary algorithms. That is not to say understanding the algorithm of evolution can solve all our problems. Wolpert and MacReady put this succinctly with the no free lunch theorem Wolpert and Macready (1997). It states that the performance of a search process, averaged over all optimisation problems, will be equal to the performance of random/exhaustive search. The no free lunch theorem is not a limiting factor when designing optimisation methods; rather, it is an important caveat to consider. Generally, we are interested in designing search methods that perform well on problems with particular characteristics. The performance of a search process is therefore determined by evaluating the performance of the search on this distribution of solutions, or as Wolpert describes how well 'aligned' a search algorithm is with the distribution of problems Wolpert (2013). In the the case of evolutionary computing, how well variability is aligned with the problem structure - the evolvability of the system.

Genetic Algorithms (GAs) emulate the process of natural selection in a population Holland et al. (1). The adaptation of a solution is a consequence of the dynamic process of variation and selection over multiple generations. The iterative process of exchanging material between fit individuals and replacing lower fitness individuals moves the population of solutions towards regions of higher-fitness. In GAs, adaptive variation is implicitly controlled by the solution representation, crossover operator, mutation rate and selection criteria. It is hypothesised that these operators combine to implicitly transform the neighbourhood of a solution by conserving adapted variable combinations (building-blocks) whilst maintaining variability between them Goldberg (1989); Holland et al. (1). In doing so, the dimensionality of the search space has been reduced from individual units to combinations of adapted units. Or, rather, variability is transformed to a higher level evolutionary unit. It is hypothesised that exchange or accumulation of building-blocks is likely to improve the fitness of a solution.

An evolutionary unit refers to an entity that is subject to variation and selection. A building-block represents cooperative association between two or more individual solution units at the solution level that contribute to the quality of a solution - a higher-order evolutionary unit - that emerges from variation and selection. Variation and selection can now act on the group as an individual unit, searching in combinations of higher-order evolutionary units. Simons Simon (1969) identifies that in naturally occurring functional systems, be that biological or human-designed, e.g., multi-cellular organisms, software, engineered solutions and social organisations, share a nearly-decomposable structure. Specifically, the organisation of the global function is a nesting of lower-order components. The internal relationships of a sub-function are stronger than the relationships with external sub-functions and are therefore prioritised. The lower-order components are combined to construct a higher-order component. This is recursively applied to construct the global function, representing a function of hierarchical organisation. Therefore, in the case of nearly-decomposable functions, the system can be decomposed into different scales of complexity and organisation. Therefore, reducing the dimensionality of an solution space by the mechanism of self-organisation aligns the search process with near-decomposable problem structure.

This mechanistic view is implicit in GAs. We desire a high degree of evolvabiilty until the global optimum is found. However, as evolution proceeds in finding solutions with higher fitness, the likelihood of adaptive variation reduces . The likelihood of finding a solution with higher fitness naturally reduces if the operators for recombination remain constant. In a GA, crossover enables the exchange of higher-order evolutionary units once emerge, providing an implicit method for maintaining a degree of evolvability later in the search.

The BBH provides a thought-provoking mechanism for evolving variability to maintain adaptive variation later in the search - the evolution of evolvability (use previous experience to guide variation of what to explore next rather than performing random exploration) Altenberg et al. (1994); Wagner and Altenberg (1996). However, the BBH has not been proven beyond first generation and is further limited to very low-order (short) building-blocks, where the likelihood of disruption is small O'Reilly and Oppacher (1995); Beyer (1997). A hierarchical construction of building-blocks, where lower-level building-blocks combine to construct a higher-level building-block requires variables within a building-block to positioned locally on the solution representation Thierens and Goldberg; Thierens (1995, 1999); Watson et al. (1998).

Never the less, the BBH provides a thought-provoking mechanism for maintaining a degree of evolvability - By evolving the representation of a solution, and or the recombination operators, whilst the population evolves, the evolvability can be adapted during evolution that allows for random recombination and selection to maintain a high likelihood of finding solutions with greater quality.

The BBH encouraged the development of algorithms to explicitly identify, capture and exploit building-block structures using machine learning algorithms. Most noteable are the class of Estimation of Distribution Algorithms that use machine learning to construct a probabilistic model of a distribution of promising candidate solutions and then sample this model to generate new offspring Hauschild and Pelikan (2011). In doing so, EDAs were capable of overcoming challenges that GAs could not Harik (1998); Pelikan and Goldberg (2001, 2006) and successfully applied to a wide range of problems Pelikan and Goldberg (2003a); Aickelin et al. (2007); Ceberio et al. (2013); Santana et al. (2008). However, in doing so, these algorithms lost their connection with evolutionary theory. EDAs use a centralised model to replace recombination operators used in GAs by modelling the alliec frequencies in a population and generating new solution using random sampling. Whilst it is argued that alliec frequencies explain adaptive variation in biologiy Charlesworth et al. (2017). It does not provide a mechanism for learning form past experience.

Watson connects how evolutionary process can learn from its experiences to shape future variability Watson and Szathmáry (2016). Specifically, the evolution of the development process from genotype to phenotype can exhibit associative memory Watson et al. (2014).

In this case, machine learning models are not used to capture the distribution of alliec frequencies, like in EDAs. Rather they explicitly capture higher-order units of variation for a solution, biased by previous experience during development. The model is used to inform variability of an individual solution (adaptively reorganise the neighbourhood of a solution) instead of the variability of a distribution.

GAs emulate the dynamical process of variation and selection, inspired by population dynamics, to improve the average quality of a distribution of solutions. Crossover provides a method for reducing the variability in a population, by mixing solution with higher-fitness contributions, and mutation provides a method for slightly increasing the diversity in a population De Jong (1975). One of the limitation of a GA is designing an appropriate crossover operator

Evolvability of a solution differs from the variability of a solution. Variability is the the ability to vary - to produce an alternative solution. Of course, this is trivial via random permutation. Evolvability refers to the ability of random permutation to produce an improvement.

Evolution by natural selection is regarded as the primary source for adaptation in a population Charlesworth et al. (2017). Variations in evolutionary units, provided by the population, are differentiated by selection. Selection prioritises the units with higher fitness for the generation of new units. Adaption occurs as a consequences of the dynamical process of variation and selection over multiple generations.

In biology, natural selection is not limited to a single level of representation. Natural selection acts at multiple scales of organisation - multiple level of pheontypic representation. The evolutionary transitions in individuality West et al. (2015) is only implicitly captured by evolutionary computing

e.g. via increasing the units reproductive success. Over multiple iterations of variation and selection (generations), solutions are encouraged to move in the direction of increased fitness - a desirable characteristics for optimisation problems.

Genetic Algorithms were the first example of emulating the dynamics of evolution for solving optimisation problems.

Important that algorithms make relatively few assumptions about the nature of the problem - allowing for general application to optimisation problems.

In this thesis, we are interested in search methods that are biased towards exploiting regularities present optimisation problem to adapt the search during optimisation. Nature provides many inspirational methods for exploiting regularities that occur from a dynamical process, such as swarm intelligent methods. In this thesis, we are interested in natural evolutionary process. Natural evolutionary processes provide us with inspirational mechanisms for identifying and exploiting the structure of an optimisation problem that has unknown problem structure (black-box) to improve search. Namely the dynamical process of variation and selection. The organisation of complex organisms show a decomposition structure, arranged as multi-level construction of sub-functions suggesting an optimisation process biased towards exploiting regularities that occur due to the dynamical process. Referring back to Simon's near decomposable structures and identifying organisational regularities, in natural evolution, the stimulus is provided by variation. Selection provides a method for extracting the response information due to variation, and variation operators, such as sexual reproduction (crossover) provides a method for exploiting the information to generate new solutions. Two methods we explore in this thesis are population dynamics that exploit alleli frequencies in a population of solution and developmental dynamics that iterative update a solution by making partial changes. Of course exploiting these relationships will not always result in finding higher-quality solutions. For example, when the relationship is not present in the global optimum, as such generating solutions that exploit these relationships will not find the global optimum, or at the very worst not improve a solution (deceptive structure).

Genetic algorithms are a well-studied class of stochastic optimisation algorithms that explore the search search space using methods inspired by the population dynamics observed in natural evolution **?**. Namely variation and selection. Variation is provide by a distribution of solutions containing alternative solutions. With variation, selection provides a pressure to prioritise solutions with greater quality. In GA, crossover is a method for exchanging information between promising candidate solutions to generate new solutions. Therefore, the crossover only operator will only reduce the diversity in a population. Mutation is an additional operator that provides a method for inducing more variation into a population i.e, A single-point mutation operator provides a probability of including local variants into the search. A limitation of using a population to provide variation is maintenance of alternative solutions during optimisation. As such, there exists a research area for methods of preserving diversity in a population to overcome the challenges of premature convergence De Jong (1975).

GAs make few assumptions about the optimisation problem (black-box optimisation technique), and therefore applicable across multiple problem domains, and applicability to derivative-free optimisation problems. GAs are therefore suitable for BBO problems as they require no 'a priori' information about the problem nor require the optimisation problem to be model by a strict algebraic model (e.g. linear and quadratic programming). GAs use a population of solutions, genetic operators and selection pressure to search the space of solutions. The population contains implicit information about the current state of the search. A population is updated by using operators to generate new solutions from the material of prioritised solutions (solutions containing information that contribute to the quality of a solution) and replace solutions of lower priority. Solution prioritisation is provided by the selection pressure, generally by favouring high-quality solutions for recombination. Each population update is called a generation. Multiple generations occur to explore the search space. At each generation the population is updated using the genetic operators to increase the average solution quality of the population.

In a GA, the model used to exploit the problem structure is defined implicitly by the population of solutions and the set of variation operators used to recombine material priorities by the selection pressure. The adaptive search is provided by the prioritising solutions with higher quality for recombination. Development of a GA has been focused on improving the alignment of the dynamical process with the problem structure. The success of GA's has been attributed to the ability of selection to prioritise common features shared in high-quality solutions (variable combinations that contributes to a solutions quality) and crossover to provide a method for recombining building-block material to generate higher-order building-blocks - known as the building-block hypothesis **?**Goldberg (1989). Specifically, the dynamical process implicitly exploits regular occurring variable combinations by prioritising solutions higher-quality solutions, and therefore containing variable combinations that contribute to a solutions quality (partial solutions) during recombination. Recombination, via crossover, provides an implicit methods of exchanging partial-solution information between solution to generate higher-order partial solutions. - generating high-quality solutions from the material of lower-quality solutions. This process is repeated, creating a recursive exploitation of successively higher-order partial solutions to create even higher-order solutions. This process of exploiting lower-order solutions to construct higher-order solutions is a process of dimensional reduction of the search space. Instead of searching all possible combinations for the higher-order solution, the higher-order solutions is constructed by checking only a few combinations of partial-solutions.

The performance is therefore dependent on the available material in the distribution of solutions (maintaining alternative partial-solutions in the population) - variables assignments and the operators used for recombination, identifying partial solutions - the selection pressure, and efficiently recombining partial solutions to search for higher-order solutions GAs have therefore been developed to improve the diversity maintenance during optimisation (preserving alternative solutions). A naive way of course is to increase the distribution size. More advanced methods include selection methods, such as binary tournament selection and replacement methods such as niching. In a addition crossover operators have been studied to understand their effectiveness. Although is skepticism of the BBH providing an explanation of the adaptive capability of a GA Beyer (1997), the intuition and idea of the building-block hypothesis never the less ignited the development of algorithms to specifically exploit and search in combinations of building-block structure. Indeed, many theoretical problems that have been developed to evaluate or understand the performance of GA are constructed using building-blocks Mitchell et al.; Watson et al. (1998, 2011); Deb and Goldberg (1994); Pelikan and Goldberg (2001); Tsuji et al. (2006); Coffin and Smith (2007). The nature and variability of the characteristics that define a building-block, specifically the positional bias, made hand-engineering operators an unfeasbile method. Thus machine learning methods were used to automatically indentify building-block structures and recombine these building-blocks.

exploration vs exploitation Črepinšek et al. (2013). Difficulties in GA (booby traps) - Weise et al. (2012) Limitations of GAs include premature convergence and disruptive recombination. Convergence of a GA occurs when genetic operators can no longer improve the quality of the population. Genetic algorithms require variation in the population to search the solutions space. Recombination and replacement provides a process that reduces the variability in the population, with lower-quality solutions being replaced by higher-quality solutions. When low-quality solutions contains use-full variables combinations, then overwriting this material before exploiting this information causes the GA to converge to a sub-optimal solution, even though the information was available during the search. .Methods to overcome this issue is to increase the population size Goldberg et al. (1992) or included diversity maintenance techniques (niching) De Jong (1975); Mahfoud (1995).

Disruptive recombination refers to the ability of the crossover over operator to efficiently recombine complete building-blocks (partial-solutions) ?Goldberg et al. (1993). This requires both identifying and efficient recombination of partial-solution's. Positional biased crossover operators, such as 1-point crossover, exploit a positional bias during recombination. As such, they provide a mechanism of exploiting partial-solutions with variables physical neighbours on the solutions string. This is refereed to as building-blocks containing tight-linkage Goldberg et al. (1993); Thierens (1995); Deb (2000); Watson (2002). Positional biased crossover operators fail to exploit problem structure containing random linkage, the set of variables for a partial solution can take any locating on the solution string. However, developing position-ally unbiased variation operators, such as uniform crossover, causes large disruption to high-order building-blocks and therefore fail to efficiently recombine the partial-solutions. Thierens (1995); Harik (1998); Thierens (1999); Watson (2002, 2006). This therefore lead to the development of using machine learning methods to explicitly capture building-blocks to remove positional bias crossover.

The often cited advantage of a GA is its ability to apply across problem domains. However, a significant proportion of research is focused on improving the recombination between solutions, either via improving the representation of a solution, so that genetic operators are more suitable Salomon (1996); Rothlauf (2006) or by improving the genetic operators to improve the alignment of the search with the problem structure Potvin (1996); Herrera et al. (2003); Pavai and Geetha (2016) for each type of optimisation problem. Further complexities are introduced by optimisation problems containing infeasible solutions and generally require hand-engineered methods to overcome infeasible solutions Salcedo-Sanz (2009); Mezura-Montes and Coello (2011). Given that a popular recommendation to use GAs is due to their general and simple application to optimisation problems due to not making strong assumptions about the problem, it is unsatisfying that significant amount of research is performed into how to perform variation that is natural to the optimisation problem. learning the variation by decomposing the problem at its natural joints and applying the variation that is natural to the optimisation problem.

However, they are limited - first is the issue with crossover operators. Then is the issue with frequency and therefore diversity maintenance

2.3 Using Machine Learning to Exploit Problem Structure

One of the earlier methods for improving the evolution of evolvability is to include hyperparamters of a GA, such as mutation rate, into the solution Altenberg et al. (1994),

The types of algorithms that transpired from the identification of linkage-learning to exploit building-block structures where, Estimation of Distribution Algorithms Mühlenbein and Paass (1996), Linkage-learning Algorithms Harik (1998). and Multi-scale Search Algorithms Watson (2006); Watson et al. (2011). Each class of algorithm uses machine learning methods to explicitly capture regularities present in a distribution of solutions, there method for identifying, capture and exploiting these relationships is what distinguishes these algorithms apart.

The class of Estimation of Distribution algorithms (EDAs), that use probabilistic machine learning models to replace recombination operators in a GA, and Multi-Scale Search Algorithms (MSSA), that use deterministic machine learning models to bias the variability of a solution, are related to the approach of Deep Optimisation. They all use a machine learning model to capture associative relationships in a distribution of promising candidate solutions, and then exploit this to bias future search. They use previous experience to guide future search - providing evolutionary of evolvability.

self-referential learning process

The BBH provided an inspired direction for the development of GAs. Specifically, the utilisation of machine learning methods to explicitly capture regularities observed in a distributing of promising candidate solutions. The machine learning model is the used to adapt the search process. The type of model used and the method for exploitation generally decides the class of algorithm. For instance, the use of probabilistic models that generate new solutions from sampling the model are within the class of Estimation of Distribution Algorithms (EDAs), where as deterministic models that adapt a search operator applied to a candidate solution are withing the class of Multi-Scale Search Algorithms. Never-the-less these algorithms can be classified as Model-Building Optimisation Algorithms (MBOAs). Specifically, a machine learning model is used to explicitly exploit regularities in a distribution of promising candidate solutions to generate new solutions - thus adapting the search process. MBOAs make the following assumption about a problem: that learn-able variable combinations observed in a distribution of promising candidate solutions can exploited to find superior solutions.

The distribution of solutions provides a signal for observing the fitness function response to different variants. Variation in fitness caused by the variation in the input can then be exploited to infer structure in the problem. Selection priorities regularities in inputs that produce a good response from the fitness function. How this information is exploited is forms the fundamental design methods that therefore distinguish these algorithms are:

1. The Model Construction
2. The Model Capacity
3. The Model Exploitation

In both GAs and MSSAs exploit regularities that occur in a distribution of solutions that are produced by dynamics of coordinated perturbation candidate solutions where selection favours high-quality individuals. This the BBH refers to a type of adaptive neighbourhood search - variation at the individual level, then moves to low-order building-blocks before finally searching in high-order building-blocks. Therefore, the process provides a signal of regularities (solutions to sub-functions) that contribute to a solutions quality in many contexts (background material).

In genetic algorithms, the search space is implicitly defined by a solutions representation and the genetic operators (such as crossover and mutation). Inappropriate representations and genetic operators can induce additional local optima into an optimisation problem Rothlauf (2006). The ability of the genetic algorithm to uncover and exploit this structure is dependent on the methods for variation and selection. As such, research has focused primarily with improving and understanding the solutions representation, recombination operators (namely crossover and mutation), population initialisation, selection and replacement methods. However, aligning the search dynamics with the problem structure requires understanding the relationships between variables for a problem instance. Machine learning provides an powerful and automated way of extracting relationships from data. There, it comes as no surprise that machine learning has been used to enhance both local search and population search methods. In this section we review how machine learning has been used to enhance optimisation. In addition, this section briefly reviews related methods, in the sense of using machine learning to enhance optimisation, but differs in the way the model is being used.

The identification of which variables interact and thus form partial solutions (and should therefore be protected) is called linkage learning: learns the linkage between variables and groups of variables.

2.3.1 Linkage Learning

Linkage learning refers to the identification of building-blocks structures. Specifically, the variable combinations that contribute to a solutions quality. The idea is that by learning the linkage information, one can design better methods for conserving building-blocks during recombination and further search.

Holland et al. (1) - small fit similarities are propagated through search

Thierens and Goldberg; Thierens (1999) - breaks down when linkage is not tight

mGAGoldberg et al. (1989), GEMGAKargupta (1996) and LLGAHarik (1997) all attempts to create tight linkage in a solution to allow for recombination, that is positionally biased to operator effectively.

By reordering the linkage information to enable crossover to be more effective, LLGA - relationships become physically closer together. crossover still positionally biased, but the restructuring of the representation conserves the relationship

In doing so this showed that buildling-blocks can be efficiently recombined in the original solution representation, the linkage is random

2.3.1.1 By Constructing a probabilistic model

Constructing and learning a good probabilistic model of a distribution of solutions requires identifying the linkage information Harik et al. (1999a) . Learning a 'good' probability distribution is equivalent to learning linkage. Constructing and learning a good probabilistic model of a distribution of solutions requires identifying the linkage information Harik et al. (1999a) .

Linkage learning in genetic algorithms is the identification of building blocks to be conserved during crossover

2.3.1.2 Deterministic model of the linkage

- Linakge tree algorithm - Multi-Scale Search Algorithm

2.3.2 Estimation of Distribution Algorithms

EDAs introduced Mühlenbein and Paass (1996) EDA summary Hauschild and Pelikan (2011) UMDA Mühlenbein and Paass (1996) PBIL Baluja (1994) cGA Harik et al. (1999b) - incremental univariate EDA - probability vector

use a centralised model

Generally, a genetic algorithm consists of a population of candidate solutions (usually a vector containing an encoding that can be interpreted to represent a solution). As such the population of a genetic algorithm contains information of the search space already explored by the algorithm. Selection, crossover and mutation can therefore be interoperated as methods to exploit this information and by using selection and crossover correctly it is possible to exploit information of good candidate solutions to direct the search (next generation) towards better solutions. However, it is acknowledged that the crossover operator, such as uniform crossover, can destroy important information contained within the population such as dependencies between variables. Using a simplistic crossover operator can destroy important dependencies even though, in a population of candidate solutions, the fitter will consistently show this dependency. In such a case, large populations are require to overcome the issue and thus renders the algorithm impractical. Therefore, whilst the information of relationships that contribute to the solutions quality is present in the distribution it also requires sufficient methods to correctly exploit the correct information in the correct way.

At the same time, Baluja used a probabilistic model to represent the population of a genetic algorithm and search using the model, much in the same way as a genetic algorithm. Whilst the model was simple, and was unable to capture relationships occurring between variables. It was never the less the first example of using machine learning methods to automatically capture information from solutions and then extract this information from the model to generate new and superior solutions. In doing so, this led to the development of the class of algorithms known as Estimation of Distribution Algorithms (EDAs). EDAs use probabilistic models to replace the crossover and mutation operator used in genetic algorithms. The models are constructed to represent a distribution of promising candidate solutions – a subset of the population filtered from the total set using selection. The models are then sampled to generate a new subset of solutions that replace existing solutions in the population. The models in EDAs are tasked with both learning the probability distribution and dependencies between variables (known as linkage learning)

Estimation of distribution algorithms (EDAs) Hauschild and Pelikan (2011) are a class of evolutionary algorithms that use probabilistic machine learning models to replace crossover and mutation to explore the search space. At each generation a probabilistic model is constructed to represent a distribution of promising candidate solutions, provided by selection. Selection provides a set of promising solutions then contain variable combinations that contribute to a solutions quality in comparison the average quality in the population. A probabilistic model is constructed to capture the joint probability distribution of the promising solutions. The model captures statistical dependencies between variables from a distribution of promising candidate solutions, which in return represent dependencies that contribute to the quality of a solution. New solutions are generated by sampling the model. We refer to this as a model-informed generation: a complete candidate solution is generated using a the model. The model therefore restricts the degrees of freedom during the process of random generation and thus collapsing the dimensionality of variation between solutions informed by information from previous promising candidate solutions. New solutions generated by the model are then introduced to the population by replacement. At each generation, a new model is constructed of the distribution of promising candidate solutions. The use of a machine learning method to identify the linkage-information (dependency structure) provides a method for removing the need to develop hand-engineered recombination operators but at a computational cost of constructing and sampling a model.

The central method of an EDA is to maintain a probabilistic model to represent the distribution over candidate solutions and adjust the model based on the results of the evaluation so that it will generate a better candidate solutions in future.

Successful for a wide range of problems Pelikan and Goldberg (2003a); Aickelin et al. (2007); Santana et al. (2008); Ceberio et al. (2013); Hauschild and Pelikan (2011)

There are two main types of EDA's. A population based EDA maintain a population of candidate solutions. Each iteration starts by creating a population of promising solutions. A probabilistic model is then built for the selected solutions. New solutions are generated by sampling the distribution and the new solutions are then incorporated into the original population. (batch mode)

Incremenetal EDA's, on the other hand, the population of candidate solutions is fully replaced by the a probabilistic model. The model is then incrementally updated by sampling, generating solutions from the model, sampling their fitness and incrementally improving the model based on these generated samples. (online mode)

The fundamental idea that forms and EDA is as follows: 1. Generate an initial population of possible solutions 2. Evaluate the correctness (fitness of the solutions) 3. Select a percentage of promising solutions from the solutions 4. Generate a probabilistic model from these promising solutions 5. Generate new solutions from the probabilistic model by sampling it 6. Incorporate the new solutions into the population 7. Repeat steps 2 to 6 until fitness function is maximized

All EDA's that are to be discussed follow these fundamental steps and they only differ in the method in how they carry out these steps, mainly how they build the probabilistic model used and therefore are generally categorised in this manner

The general EDA algorithm is presented in Algorithm 1. The fundamental idea is to

Algorithm 1: The general Estimation of Distribution Algorithm

```
Initialize: Population of Solutions;
while Termination Criteria not met do
    Evaluate the fitness of each solution Select a set of promising solutions from the
      population Construct a probabilistic model that estimates the distribution of
      the promising solutions set. for N in NewSolutions do
        Sample the probabilistic model to generate a solution Replace a solution in
          the original population with the generated solution
```

maintain a probabilistic model to represent the distribution over candidate solutions and adjust the model based on the results of the evaluation so that it will generate a better candidate solutions in future. The model is either reconstructed (constructed from no information) in update (incremental). The majority of methods apply the former approach. The general design decisions for an EDA are:

- The selection method for creating a distribution of promising candidate solutions.
- The Model used to capture the probability distribution - this impacts both the model construction and sampling methods
- The replacement method for incorporating a new solution into the population

2.3.2.1 Univariate Models

The earliest examples of EDAs used simple uni-variate models to represent the joint-probability distribution of solutions Baluja (1994); Mühlenbein and Paass (1996); Harik et al. (1999b) and was an important step into understanding how machine learning models can be used to in the framework of genetic algorithms - specifically, for capturing statistics of a population and exploiting these to generate new solutions of greater quality. However, due to using only uni-variate statistics, they were not sufficient to capture relationships between variables and therefore not identify variable combinations that contributed to a solutions quality (building-blocks).

- generating a new solution for a probability vector.

The Population-Based Incremental Learning (PBIL) algorithm that replaces the population with a probability vector that is updated via incremental updates to the model. The compact Genetic Algorithm (cGA) (Harik Goldberg, The compact genetic algorithm , 1997) generates the probability vector by generating two potential solutions and select the solution with highest fitness. The variables that differ in the winning solution in comparison the loosing solution have their corresponding probability updated towards the winning bit value. Like PBIL the cGA is an incremental EDA and provides advantages with reduced memory requirement. However the limitations of all the uni-variate models is the inability to solve complex problems due to the assumption that all variables are independent.

In EDAs, deciding which variables conditioned on other variables, or determining the structure of the probabilistic network, is referred to as linkage-learning. Thus linkage-learning only refers to which variables show relationships, and not the value of these relationships (i.e., the conditional probabilities). By capturing and conserving linkage during recombination allows for minimising the disruption to building-blocks during recombination.

2.3.2.2 Bivariate Models

This therefore drove the development in this field towards using more sophisticated models. The next phase introduced bi-variate models into the framework. Bi-variate models form dependencies between variables in the form of a chain De Bonet et al. (1997), tree structure Baluja and Davies (1997) or a forest (mutliple trees) Pelikan and Mühlenbein (1999). By sampling the model to generate new solutions, the linkage information between a pair of variables is conserved, allowing for an improvement when performing building-block recombination.These model showed an improvement in the performance on decomposable problems compared to uni-variate model EDAs due to capturing pairwise relationships in the sub-problems. However, these models are limited to only decomposing the model to order two and therefore development continured to form an implementation using a multi-variation models.

2.3.2.3 Multi-Variate Models

The most sophisticated models used by EDAs are capable of multi-variate factorisation of the joint-probability distribution. This allows for high-order variable relationships to be captured by the model. However, the cost of constructing the models become more complex and computationally expensive.

Factorised Distribution Algorithm (FDA) Mühlenbein and Mahnig (1999)- uses a fixed factorised distribution as the model and as such the structure is not learnt during optimisation. Therefore requires prior knowledge of the problem structure.

The extended Compact Genetic Algorithm (eCGA)Harik (1998) constructs a marginal product model that groups variables into independent sub-sets (variables with in a subset are dependent). A greedy algorithm combines two groups together that increase the Minimum Description Length (MDL) for the distribution model. Initially each variable is considered an independent group. When no grouping can increase the MDL metric, the model construction is terminated. The probability for each group is calculated from the statistics measured in the promising candidate solution for the group. The probabilities are then sampled to generate new candidate solutions. Therefore, the groupings will represent building-blocks and the random sampling allows for recombination between building-blocks. However, with this model, overlapping building-blocks cannot not be accurately represented.

The Bayesian Optimisation Algorithm (BOA) Pelikan et al. (1999) uses a Bayesian network to model the distribution of promising candidate, where a node represents a variables and an edge represents the conditional dependence between nodes. The Bayesian network model, an acyclic graphical model, allows for a variable to be conditioned on the value of multiple parents (unlike a tree structure).

A scoring function and heuristic search are used to construct the model due to the complexity of constructing a Bayesian Network being an NP-hard problem. BOA uses a greedy hill-climber and the K2 metric Cooper and Herskovits (1992). There is a significantly cost for the model construction algorithm as for each modification made to the Bayesian Matrix (addition, removal or reverse of an edge) the metric needs to be calculated. BOA reduces the computation cost by only considering edge additions to the network. Never-the-less the time-complexity for constructing the Bayesian network scales exponentially with the maximum number of incoming edges (k) to a node in the Bayesian network as Np^k where Np is the number of problem variables. Thus in the BOA, k is a tune-able parameter, with higher k allowing for higher-order dependencies to be capture by the model.

The BOA works by initialising a population of candidate solutions from a uniform distribution of possible solution states. The fitness of individual solutions is evaluated. Truncation selection is then applied on the population to filter-out promising solutions which are then used to construct the probabilistic model (generally a 50% separation for model-building and replacement. From the model, new solutions are generated using forward sampling. These solution replace the sub-set of solutions from the population that have the lowest fitness. This process is repeated until the termination criterion is met. BOA is capable of learning overlapping dependencies, unlike ecGA, and the BOA does not require prior definition of the model structure, unlike the FDA.

2.3.2.4 Hierarchical Bayesian Optimisation Algorithm

The most sophisticated, and considered state-of-the-art is the Hierarchical Bayesian Optimisation Algorithm (hBOA) Pelikan et al. (2003); Pelikan and Goldberg (2006). hBOA is an extension of the BOA. The first extension is the use of a decision trees to represent the Bayesian network model. In BOA, the number of conditional probabilities grows exponentially as the number of interactions between variables grows. hBOA overcomes this by exploiting regularities in conditional probabilities and encoding these using a decision tree. This representation allows for the model to include high-order interactions more efficiently. hBOA uses a forest of decision trees to achieve this, with each variable containing a decision tree that represents the variables that condition the probability for the variable.

The metric used to measure the performance of the model during construction is based on pairwise statistics. A greedy algorithm is used to add edges between nodes that improves the performance score. The metric used during construction contains an implicit inductive bias towards simplier models

hBOA maintains a population of candidate solutions and applies binary tournament selection to produce a distribution of solutions that have better than average quality given the population. This distribution of promising candidate solutions, containing variable combinations that contribute to the solution quality, is used to construct a Bayesian model. New solutions are generated using forward-sampling and replace other solutions in the population using restricted tournament replacement (RTR) Harik. Generally, half the population is used for constructing a model, and the same number of solutions are generated from the model. The number of replacements is determined by RTR. This process is repeated with a new Bayesian model created each time, no information in a Bayesian model from earlier generations is retained.

RTR causes competition between a newly generated solution and the existing population to be similar - prioritises competition between similar solutions rather than at random.

A window size defines the size of the niche - a window size the size of the population will ensure that the nearest solution in the population is used for competition

hBOA is used through-out this thesis as a state-of-the-art EDAs. It has proven to be effective at solving a wide range of optimisation problems Pelikan and Goldberg (2003a); Pelikan (2010) and is often used as a state-of-the-art example when developing new algorithms Thierens and Bosman (2013); Goldman and Punch (2015); Hsu and Yu (2015)

2.3.2.5 Neural Network Models

Artificial neural network models (NN) have provided state-of-the-art results in domains such as classification, regression and generative tasks. A NN consists of an input, hidden and output nodes that are connected via weights. The network computes an output from an input. The function the network computes is trained. Greater detail about NN models is provided in Chapter 4.

Neural network models provide advantages over other machine learning models used by the state-of-the-art EDAs. They are flexible, high capacity models that use efficient learning algorithms to construct to approximate a function. In addition, NN models used hidden variables (latent variables) for representing relationships between that compute the output of the network. It is known in Bayesian network models, the inclusion of hidden variables for representing dependencies between variables reduces the complexity of the network when capturing high-order relationships Elidan et al. (2001). A disadvantage of NN models is the black-box nature, where understanding what the model has learnt is an open and ongoing area of research.

The idea of using a neural network as the model an is not novel. Santana Santana (2017) provides a brief but excellent overview of the algorithms that use a neural network within the framework of EDAs. However, in the domain of EDAs, and more generally using learning to improve optimisation, neural network models have failed to provide state-of-the-art results Churchill et al. (2014a); Probst (2015). Considering state-of-the-art performance is achieved using a NN in other domains were alternative machine learning models, such as the Bayesian Network, are applicable, it is somewhat surprising for a NN to show inferior performance.

In general, EDAs that use a neural network have are generally limited to a single layer network Probst (2015); Churchill et al. (2014a); Santana (2017) and proceed in the typical framework of EDAs, where the NN is sampled to generate new solutions.

A hypothesis for why SOTA performance has not been achieved is due to the limitation of the depth of the NN. A deeper network provides additional capacity to the model and allows for abstraction of higher-level features from the data-set. However, little research has focused on using a deep NN. Examples of algorithms that use a deep neural network are the Deep Boltzmann Machine Estimation of distribution Algorithm (DBM-EDA) Probst and Rothlauf (2015) and the Deep-Opt Genetic Algorithm Baluja (2017). For DBM-EDA, the model is

DBM-EDA a deep Boltzmann machine (DBM) with two-hidden layers as the model. The DBM is trained to learn the probability distribution of the selected candidate solutions and then sampled to generate new solutions. In Deep-Opt GA, the model is used to approximate the fitness function. New solutions are then generated using a technique called network-inversion where the back-propagation algorithm is used to modify the inputs into the network rather than the weights (as done during training). Whilst not technically inline with a the EDA framework as the model captures relationships between input variables and the fitness values, rather than just the input variables, it is never-the-less sufficiently related and a rare example of using a deep neural network to improve evolutionary search. In both cases, there is no evidence for understanding why and if a deep network provided a performance improvement to the algorithm compared to a shallow model. Therefore an important question remains, what type of problem structure can a deep NN model capture that a single layered (shallow) NN cannot.

Understanding how to utilise a NN model and exploit a deep representation remains an open and promising area of research. The main design decisions is for how to train the neural network model, i.e., what information the network is modelling and how to exploit this information to improve the candidate solutions.

A significant advantage of a NN model compared to other machine learning models used by EDAs is its flexibility. For instance, an Autencoder model can be used in a variety of ways and interpreted as a probabilistic model that captures the distribution of a data set or as a deterministic dimensional compression of the dataset. Therefore, a NN advantage also is its disadvantage as it opens up many questions for how to use to model, in terms of both training and exploiting the information. Much of the literature has remained faithful to the EDA framework.

Performance outside of optimisation has shown that deep models provide can learn meaningful structure to provide SOTA performance.

Neural network models provide a method or representing the problem structure using latent variables. The design decision therefore open are how to exploit the information available in the latent variables. In Deep Optimisation this is achieved by interpreting the latent variables as a transformed and compressed solution space of the layer below.

Probst - computationally fast when compared to constructing a Bayesian network. but not as good as power

advantages and disadvantages listed by Santana:

- Flexible Models - Able to model complex dependencies - multivariate models(5)
- Efficient learning algorithms (25, 121)
- natural and proven parallelism via GPU (26)
- naturally applied for transfer learning (25)

Disadvantages:

- sampling can be cumbersome
- NN sensitive to initial parameters used for training
- Representation of the latent space, doesn't always correspond to representation of problem
- large number of parameters for learning
- over fitting

neural network models are less restrictive that the models using in EDAs, and therefore the methods for exploiting the model information generally vary between the work

A limitation is their 'black-box' nature and therefore causes complexity when attempting to understanding and interpreting the models performance.

Probst (2015)

number of hyper parameters associated with these models. Therefore, in optimisation,

However, evaluation of there performance has failed to produce state-of-the-art results.

highlight that deep network models can provide significant advantages however attempts thus far have not been competitive va let alone outperforming other state-of-the-art methods. is there a reason that this is not just due two learning The relationships But also so so efficiently and accurately exploiting relationships.

but designing and understanding methods that use deep learning have proven unsuccessful and thus far - is that because we are unsure how to utilise the depth of model

2.3.3 Multi-Scale Search Algorithms

Natural selection acts on the phenotype, the variability of a phenotype is biased by previous experience

Multiple levels of phenotypic representation - Deep Optimisation

A less well-known class of optimisation algorithms is Multi-scale search algorithms (MSS). MSS algorithms are inspired by the developmental dynamics observed in natural evolution to provide adaptive variation Watson and Szathmáry (2016); Uller et al. (2018). Specifically, a solution is updated using variation operators that are re-scaled to successively higher-orders of organisations. The variation operators are constructed from relationships observed in previous solutions that contributed to a solutions quality. A fundemental difference from GAs, is that the indivudal fittness is prioritised.

Efficiency due to accepted solutions during update containing implicit information about the solution space Incorrect partial solutions are immediately rejected due to selection neglecting updates that provide a deleterious change.

Associative relationships such that solutions that are were previously far away become closer. - model space

Multi-Scale Search Algorithms (MSSA) are an alternative approach to providing a source of adaptive variation inspired by the BBH and evolutionary processes. Like EDAs, they use machine learning models to capture building-block structures from a distribution of candidate solutions and exploit this information to improve the search process. However, they differ namely in the methods used to exploit information during search. In EDAs, search is preformed by random recombination of partial solutions - the model is used to generate complete candidate solutions. In MSSAs, search is explicitly performed at the scale of building-blocks - the model is used to rescale the search operator to higher-orders of organisation Iclanzan and Dumitrescu (2007); Watson et al. (2011); Watson (2006); Mills and Watson (2011); Mills et al. (2014). We can therefore describe the idea of MSSA as adapting a local-search method to search in the space of building-blocks.

Therefore, an important distinction between EDAs and MSSAs is that in an EDA the performance of a population is prioritised over the performance of an individual (although this is not strictly true in hBOA). In the case of a MSSA, the performance of an individual is prioritised. In EDAs, promising solutions provide information about the natural structure of an optimisation problem. MSSA make the same assumption, however, the set of promising solution is determined in an alternative way. In general, for EDAs, solutions that have above average fitness (by rank) are considered as the set of promising candidate solutions. However, in MSSA, each solution, that has been undergone local-search is considered a promising candidate solution. This has implications that are discussed in Chapter **??**.

A MSSA can be interpreted as a repeated local search algorithm that adapts the search neighbourhood according to observed relationships from previous attempts (its own dynamical experience). Thus, at initialisation, the search operator is naive, and allows for a natural exploration of the search space, e.g. for a binary solution a 'bit-flip' operator can be suitable and its performance is equivalent to a local search algorithm. Local search then proceeds in a new, redefined neighbourhood where variation of solution constitutes to simultaneous changes to multiple variables rather than, the original lower-order operator of, single variables. Thus, the algorithm adaptive reduces the dimensionality of the search space and is biased to exploiting regularities that occur in a distribution of locally optimal solutions found in the higher-dimensional search space.

Local Search (LS), provides an efficient method for optimisation by prioritising only local alternatives during search. A solution is updated on the basis of selection criteria and a neighbouring solution. For a hill-climber, the selection criteria is based only on the solution quality. Solutions outside the local neighbourhood are not considered. The neighbourhood is repeatedly checked for superior solutions and if no neighbouring solution has a greater fitness than the current solution the algorithm is said to have converged. First-improvement hill-climbing refers to an interactive procedure that randomly selects a single neighbouring solution. The solution updates the current solution if it has a greater solution quality. Greedy local search evaluates all neighbours of the current solution and updates it with the solution that has greatest solution quality.

LS is biased to exploiting local information about the problem structure (search space). The performance of LS is strongly influenced by the current solution and the neighbourhood of the solution. Therefore, unless the problem contains a single attractor, or the initial solution is in the correct basin of attraction for the global solution, local search will fail to find the global optimal solution. Thus the main weakness of a LS method is that they become easily trapped at sub-optimal solutions, where no neighbouring solution has a greater quality that the current solution yet withing the possible set of solutions there exist a solution with greater quality.

The success of local search is therefore dependent on the alignment of the neighbourhood and the problem structure. Of course this requires understanding the problem structure. If one can learn and understand the problem structure of the optimisation problem, one can develop local search methods to exploit the problem structure. For well-behaved problems, where regularities in the problem structure exist for a large sub-set of problem instances ,this can be particularly effective and worth the human effort to develop methods. A successfully example is the Lin-Kernighan heuristic for the symmetric travelling salesman problem Lin and Kernighan (1973). By developing local search methods to exploit the problem structure, it is possible to efficiently search in the solution space of a CO problem.

Alternative methods to overcome the weakness of LS becoming trapped at sub-optimal solutions include restart, or multi-start, methods. Restart methods use a single solution instance. The solution is updated using local search. After local search terminates, the solution is reinitialised. Multi-start local search uses multiple solution instances, with each instance being update using local search. There is no performance difference between the algorithms. Rather multi-start local search provides a distribution of solutions that are at an attractor of the solution space. Importantly for MSSA restart methods allow for efficient exploration of multiple basins of attractors in the solution space. However, these methods use no information about previous solutions.

Examples of methods that do use information about the solutions found thus far are Iterated Local search (ILS) and Tabu Search (TS) Glover (1989) and Simulated Annealing Kirkpatrick et al. (1983). However, the information exploited is only based on memory, either the best solution found so far, or which solutions have been already explored. MSSA on the other hand work by learning and exploiting the realtionships between variables that contribute to a solutions quality.

MSSAs share similarities with the well-known Variable Neighbourhood Search (VNS) method Hansen et al. (2010). VNS work by performing local search if a fixed set of neighbourhoods that are manually defined prior to searching in them. However, in MSSA, the idea of search in a new neighbourhood is shared with VNS, but MSSA instead adaptively reorganises a solution neighbourhood by capturing relationships from a distribution of solutions found in the current neighbourhood.

MSSA induce a new and intelligent space to search in by capturing relationships found in a distribution of solutions that have been hill-climber and therefore locally optimal relative to the current solution neighbourhood. The local optimal solutions provide information about the natural basins of attractors and specifically the relationships between variable combinations that contribute to the solutions quality. A model is used to learn and capture these relationships. The information is then used to transform the search operator to higher-orders of organisation, effectively transforming the neighbourhood of the solution such that partial solutions that were not local in the original neighbourhood become neighbours. Search then continuous using this neighbourhood, either by reinitilising a solution Watson et al. (2011) or updating the exisiting population Iclanzan and Dumitrescu (2007).

The advantage of a LS is that it provides an efficient method for locating basins of attractions in the solution space. MSSA use the information contained in these basins of attractors to addaptively reorganise the solution landscape such that basins of attractions become neighbours at the new search space - an effective dimensional reduction of the search space Watson et al. (2011). An attractor, in terms of optimisation problems, defines a solution in the solution space that for a large number of solutions will tend towards. Therefore, we can interpret this as solutions within the basin of attraction will regularly lead to this basin. Thus we can assume that the possible set of variable combinations within basin of attractor can be compressed to the correct sub-set of variable combinations - reducing the dimensionality of the search and thus subsequently adapting the neighbourhood of the solutions space. The attractor is defined by the relationships between variables that contribute to a solutions quality. The attractor's provide information about regularities in the search space that contribute to a solutions quality. By modelling the information of fixed point attractor provides a method of dimensional reduction of the solution space Watson et al. (2016). The reorganisation of the neighbourhood, via exploiting correlated variables, changes the effective fitness landscape for the solution. Specifically, fitness valleys between between these points in the solutions space are no longer present - a new fitness landscape is presented to the solution, as defined by the new neighbourhood and the fitness differences between these two points. The effect on the fitness landscape is a course-grained representation. For instance, we can reduce the dimensionality of the search space as we can infer that all solutions in the basin of attraction will lead to the attractor. Reducing the dimensionality of the solution space by transforming multiple variables into a single higher-order unit. Search then continues using the higher-order units to explore the solution space. Searching at this level of representation provides a method for escaping local optimal solutions, by allowing larger changes to a solution and to follow the fitness gradient observable between basins of attraction as illustrated in Figure 4.1. At a higher-level, many small basins of attraction can be withing a larger basing of attraction which can be easily followed when the algorithm is capable of searching in the higher-order units of partial solutions Watson et al. (2011); Mills et al. (2014).

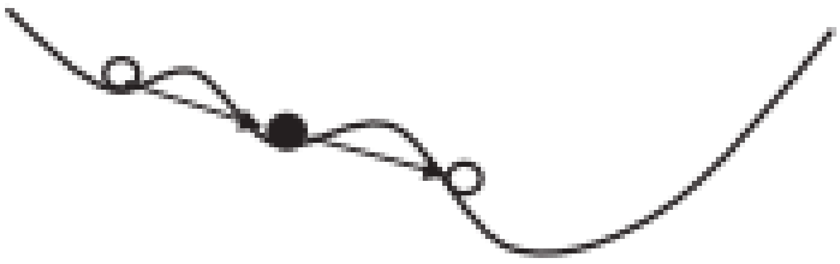


Figure 2.2: Problem structure for the manifold theoretical optimisation problem. The gradient is simple to follow if an optimizer can model and separate the directions of variation and hill-climb

Reducing the dimensional of the search space is achieved forming coordinate relationships between variables and appropriately separating them at the natural joints to form interchangeable building-blocks. This reduction in dimensionality of a search space is directly linked with the theorem of why GAs can perform well using crossover. The Building-Block Hypothesis states that variable combinations that contribute to a solutions quality are sampled and recombined to generate solutions with higher solution quality. Thus search is performed at a higher-level of organisation, search has adapted from searching each dimension individual to searching in variable combinations. In a GA, the exploitation of building-blocks is implicit in the crossover operator. Therefore, in order for recombination effective, the recombination operator - being crossover in GAs or the sampling method in EDAs, must conserve the building-blocks during search. In

MSSA, building-blocks are explicitly exploited by searching in the space of building-blocks. Searching in the space of building-blocks can be interpreted as re-scaling of the search operator. For example, the search operator is a single-variable substitution at first, then, after the identification of partial-solutions, the search operator is adapted to searching in partial-solutions - a higher-order search operator.

The general MSSA is presented in 2. A distribution of solutions is used and a model is either incrementally updated by each solution (online learning method) Mills (2010); Watson et al. (2011); Cox (2015) or constructed using a distribution of solutions (batch learning) Iclanzan and Dumitrescu (2007). At first, the algorithm searches in the space of the primitive units to identify structural correlations. It is important for the performance of the algorithm that search provides a natural and unbiased exploration so that misleading correlations are not induced. This is then used to redefine the search operators of the search space. A model is then used to capture the relationships between variables in the distribution of solutions that are locally optimal in the current solution neighbourhood. The model is then used to reorganise the solutions neighbourhood and each solutions in the distribution is updated by performing local search in the new redefined neighbourhood. A solution is optimised by iteratively updating the solution using partial solutions - each attempt of changing or including a partial solution to the current solution is checked via selection. Thus adaptation of the search operator (or solutions neighbourhood), via learning associative relationships between variables, adapts the partial solutions applied to the solution, and therefore the neighborhood of solution.For a binary optimisation problem a local search that is often used to search in the local neighbourhood is called a 'bit-substitution': a random variable of the solution is replaced with the variable value with a 1 or a 0. The search method is the lowest-level variation operator available and allows for all possible solutions to be explored.

The principle algorithm is similar to the EDA algorithm. A model captures the relationships observed from a distribution of candidate solutions. Each solutions in the distribution is update by local search and therefore all solutions in the distribution are considered promising candidate solutions. The fundamental difference from EDAs is that the model is used to transform the neighbourhood of a solution. Instead of generating a random combination of building-blocks, the search operators of a local search algorithm are rescaled to higher-orders of organisation providing a method of combining and evaluating each building-block separately. Like EDAs, the algorithm repeats the cycle of improving solutions and learning information from these solutions to improve the next iteration - the model captures higher-order regularities at each iteration. The original high-dimensionality of the problem is reduced to progressively lower-dimensionality by adapting the search operator to successively higher-orders of organisation

The solutions found at after each iteration are a product of the own dynamical behaviour of the algorithm. If the distributions of solutions is not sufficiently to allow identification of problem structure, or the model is unable to capture the relationships between variables, then the induction of spurious relationships are possible in the model. Exploiting this information can therefore lead to the algorithm converging on inferiorior solutions. Thus the performance of the algorithm is dependent on its own dynamical behaviour

Algorithm 2: The general Multi-Scale Search Algorithm

```
Initialize: Model;
while Termination Criteria not met do
    Initialize: Solution;
    while Optimising Solution do
        Make partial change to solution, defined by model;
        if Partial Change improves solution quality then
            Accept partial change to solution;
        else
            Reject partial change to solution;
        Update the model using optimised solution;
```

In comparison to EDAs, there are relatively few examples of MSSA. However, the design decision for MSSA methods generally relate to initialisation of a solution, the model used to capture relationships between variables and the methods for the redefining a solutions neighbourhood by exploiting the information from the model.

The Building-Block Hill-Climbing algorithm BBHC Iclanzan and Dumitrescu (2007) is one of the first examples of a MSSA. It performs local search on a distribution of solutions that are locally-optimal before then using these solutions to construct a model, representing a bijective mapping from building-block to variable i.e., which variable belongs to which building-block. The configuration of a building-block (the values of the variables in a building-block) are extracted from solutions that have undergone hill-climbing. Thus, the model in BBHC learns the linkage and the variable assignments for search. Hill-climbing in the space of building-blocks is then performed by selecting a building-block and evaluating the change in the solutions quality for each possible configuration of that building-block (model-exploitation). The configuration that provides the greatest increase in a solutions quality is then kept. This is repeated on all solutions before a new model is then constructed. Whilst the BBHC provde efficient compare to EDAs, the **something about complexity of learning and model - cannt recall** and the BBHC makes no attempt at biological plausible mechanisms for evolution.

Watson however has provided a connection between MSSA and biological evolutionary processes, namely the evolution of the developmental process Watson et al. (2011, 2016). Watson's recent connections in evolution and learning Watson and Szathmáry (2016) suggest that the development process is able to facilitate a type of learning that can exploit information from previous experiences. This is acheived by a separation of time scales between updating a solution (local search) and updating the model. facilitates a mechanism of evolution updating the developmental bias of developmental process of an organism. The developmental bias provides a method for learning correlation between variable than enables a transition from updating individual units simultaneous updates to multiple variables Watson and Szathmáry (2016). The interaction between the inner and outer loop is connected with with the theory of evolving the developmental process in biological evolution, and specifically the separation where the update made by development is performed multiple times, or has a stronger effect, than the update made by evolution. In biology, development facilitates adaption within the life-time of an organism. At the evolutionary time scale however, adaptation occurs over multiple generations, i.e., many development cycles. This is capture in MSSA by controlling the learning rate in online, or controlling the distribution size in batch mode.

The space for which development acts on is evolved over time, from individual evolutionary units, that form associative relationships to form higher-order evolutionary units that construct building-blocks. The Evolutionary Transitions in Individuality (ETIs), observed in biology West et al. (2015), provide an additional inspiration to develop evolution algorithms. Specifically, evolutionary processes operating over multiple levels of organisation. 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 Smith and Szathmáry (1997); Watson and Szathmáry (2016). For example, unicellular life transitioned into multi-cellular organisms — and these are not merely cooperative relationships among co-evolving unicellular organisms but new evolutionary units. These new units allow combinations of units from the previous layer to be Each subsequent layer re-codes again, in the analogue of successive hierarchical transitions in individuality Watson and Szathmáry (2016).

Evolutionary search using evolutionary units - an evolutionary unit is adapted over evolutionary time. At initialisation, the an evolutionary unit corresponds to the lowest-order building-block - an individual variable of a solution. Over time, higher-order evolutionary units develop from the a combination of low-level evolutionary units - thus a higher-order evolutionary units is a transformation of a combination of low-level units. At this higher-order representation, the higher-order evolutionary is used during search allowing for simulatenous change to multiple variables before selection acts - the same process that acted on the low-level evolutionary units. Thus, in MSSA, the adataptie response is a product of coordinated restructuring of the search operator, or neighbourhood space of a solution.

In relation to searching for a solution to an optimisation problem. Development provides a method for updating a solution within lifetime and its search is biased by the units it acts up. Initially, these units are at their most primitive - individual solution variables. Over time, these unit for associative relationships transforming the set of units into a coordinate unit that variation and selection can act-upon. In evolution, this is referred to as evolutionary transitions in individuality.

Examples of algorithms that demonstrate this connection are the Schema Grammer Cox (2015); Cox and Watson (2014b), MACRO Mills (2010); Mills et al. (2014) and rHN-G Watson et al. (2011). They perform an online adaptation of a neighbourhood Mills (2010); Watson et al. (2011); Cox (2015) by making incremental updates to the model after each solution has been optimised.

In this thesis, we concentrate on the algorithm of rHN-G due to its connection with developmental process, used of a simple correlation learning model and efficient model exploitation and ability to capture hierarchical structures. rHN-G is used as a predecessor for the development of Deep Optimisation.

rHN-G Watson et al. (2011) can be described as a random-restart hill-climbing algorithm which contains an inner-loop and outer-loop. The inner loop is a hill-climbing algorithm that performs optimisation of a solution. The outer outer loop resets the solution using a random initialisation and updates the model using the optimised solution to adapt the neighbourhood of the next solution. As rHN-G is an online approach, the restart allows for exploring different basins of attraction in the solution space of the problem.

The reorganisation of a solutions neighbourhood is achieved by creating associative relationships between variables via Hebbs rule - 'what fires together wires together'. The strength of the association between variables is then used to determine whether variables form a higher-level unit during variation. Specifically, if a single variable is to change, what other variables should also change and in what direction - positive or negative correlation with the variable. The probabilistic threshold of the strength of associates is used to construct a partial solution to substitute into the solution that contains the variable selected to substitute, thus rescaling the search operator to higher-orders of organisation. The inner loop proceeds as normal, accepting changes that make an improvement to a solutions quality.

As the name suggests, rHN-G is based on the dynamical behaviour of a Hopfield network Hopfield (1982). By encoding an optimisation problem into the connection matrix, the Hopfield network will converge to point attractors in the optimisation problem. By observing the dynamics of the Hopfield network as it converges to point attractors, the algorithm is able to learn the underlying structure to the optimisation problem with simple associative learning mechanisms. In the learnt model, each variable is connected to all other variables describing a complete undirected graph or correlation matrix. The weights between discrete states form a connection matrix and its values come to reflect the constraints of the optimisation problem to be solved. For an optimisation problem that has constant self-weights ($\omega_{ii} = 1$), symmetric constraints ($\omega_{ij} = \omega_{ji}$) and uses the energy function in Eq. 2.1, it can be guaranteed that the network will find locally optimal solutions of the fitness landscape given sufficient time to relax Hopfield and

Tank (1985).

$$E_S = -\sum_{ij}^{N} \omega_{ij} S_i S_j \tag{2.1}$$

In replacement of using the Hopfield equation of motion to relax the system, rHN-G deterministically performs updates to the Hopfield network: equivalent to a gradient descent algorithm. Here a variation operator is applied to the current state:

$$S'(t) = \{S_1, S_2, \ldots, -S_X, \ldots, S_N\}, X \sim U(1, N) \tag{2.2}$$

If the change in state causes the total energy to reduce then the state is kept: $S(t+1) = S'(t)$, otherwise the state is rejected: $S(t+1) = S(t)$. Thus, rHN-G updates the state by stochastically changing variables in the state only if it decreases the system's energy. This is a process conventionally used in stochastic local search and mutation based evolutionary algorithms. Given sufficient number of iterations of this energy minimisation process, the solution will converge to a local optimum present in the fitness landscape. rHN-G then applies Hebbian learning HEBB (1961) to update its pairwise correlation matrix M which represents the observed correlation frequency from past potential candidate solutions. As such, learnt matrix M can be regarded as an associative memory model of the local optima of the optimisation problem. Initially the correlation matrix $M = 0$ and is update after each constructive selection process, T (time to reach a local optimum or a pre-determined number of iterations):

$$m_{ij}(T+1) = \gamma[m_{ij}(T) + \delta S_i(T) S_j(T)] \tag{2.3}$$

where γ is a linear threshold function capping the learnt correlation strength to a magnitude of 1 and δ is the learning rate. Given that the locally optimal solutions contain information about the problem structure it is then possible to learn the underlying structure of the original optimisation problem.

rHN-G starts with a randomly initialised solution state. After relaxing the Hopfield network, Hebbian learning is used to update the correlation matrix. The solution state is then reset (restarting the Hopfield network solution) to a random solution state from a uniform distribution and the process iterated until the global optimum is reliably found. Resetting the solution state to a random configuration allows for exploration of the solution space necessary to learn its structure. By remembering information from each locally optimal solution that is visited, rHN-G is able to learn the optimisation problem structure.

2.3.3.1 Using the model to identify partial solutions

What differentiates rHN-G from other methods is the way that selection is used in the process that generates samples from the learnt model. Specifically, it uses the correlation matrix to (probabilistically) perform a state change to multiple state variables simultaneously (creating a new search process at a higher scale of organisation) and interleaves this with selection. This can be interpreted as modifying the variation operator to perform a block substitution instead of a single-bit substitution.

A block is constructed probabilistically from the learnt pairwise correlation matrix M. A discrete variable is first selected, S_X; as the decision variable of the block that will be created, S_B. The probability of other discrete variables, S_j, being included in the block is determined by the strength of the correlation with S_X . For each block-substitution, a random number, r, is generated from $U(\mathrm{E}[M], 1)$ which sets the threshold correlation strength required to be included in the block. As $M_{ii} = 0$, the identity matrix is added to M so that at least a single variable is changed in each substitution (the correlation matrix for a single-bit substitution simply being the identity matrix). If the strength of the learnt correlation is greater than the threshold limit r, then the discrete state S_j is added to the block otherwise the state variable does not change value. The sign of the correlation determines if the state agrees (positive) or disagrees (negative) with the parent state once substituted. This generates the new state variable S' and is selected if it increases the fitness, hence an example of constructive selection.

Initially, before the correlations have been learned, rHN-G will act as a simple-hill climber performing only a single-bit substitution for generating new samples. As it learns from the distribution of local optima that it samples it will transform the single-bit substitution operation to perform block-substitutions, increasing in size over time. Hence rHN-G rescales the search space to lower dimensions (searching combinations of partial solutions/blocks).

The model is not being used to generate complete candidate solutions on its own, as BOA does, rather the model is being used to generate partial solutions. The process of creating a complete candidate solution involves an iterative loop of substituting these partial solutions into the full solution. Selection is used to determine whether the full solution fitness has been improved and if so the partial solution is retained, i.e. constructive selection. The evolution of a solution state for an MC problem shown in Fig. 2.3 shows this behaviour where partial solutions are replacing the full solution in order to provide better fitness.

Appling the MC problem to rHN-G, one can clearly see that the connection matrix in rHN-G is equal to the constraint matrix in the MC problem, with the energy function being equal and opposite to the fitness function (minimising the energy finds higher fitness).

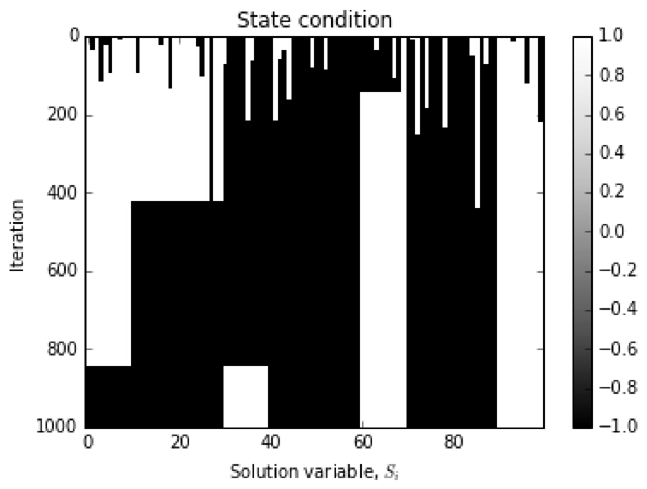


FIGURE 2.3: Example state (solution) configuration during one constructive selection process on an MC problem ($N = 100$, $n = 10$) after learning from 15 training examples. This shows that block-substitutions that improve fitness (changing multiple solution variables simultaneously) can occur because they are informed by the model.

rHN-G Watson et al. (2011) - online learning - separation of time scales - most related to DO

rHN-G works in online mode and uses generative associations between variables to transform the neighbourhood - the neighbourhood is problematically determined.

Results show that MACRO is more efficient than BOA Mills et al. (2014) - in Chapter we show that rHN-G can solve problems BOA cannot.

rHN-G differs from the BBHC as it uses a probabilistic interpretation of the correlation strengths to determine the partial solutions. In doing so, this allows for a hierarchical representation via differences in correlation strengths between variables. As shown in Chapter 6, rHN-G is capable of representing hierarchical structures by a difference in the strength of the correlations between variables. However, the model is limited to non-overlapping structures. Mills Mills et al. (2014) described this difference as soft-joints and hard-joints. Hard-joints make discrete and explicit joins between variables such that higher-order unit is a discrete representation of the lower-level variables (there is no intermediate representation, the strength of the dependencies between the variables is the same for all variables involved.). Where as soft-joints are interpreted problematically and therefore a building-block is never discretely represented by a higher-level units, it is rather problematically determined, with the strength of the dependencies between variables used to define likelihood of a variable contributing to a building-block.

Summary of MSSA - why we are interested in MSSA, how this is linked to Deep Optimisation. In all this, MSSA uses relatively simplistic models compared to EDAs. Thus their capacity to represent relationships between variables is inevitably lower than those used by the SOTA EDAs. One notable difference is the abilty to capture overlapping dependencies betwee variables. However, results show an efficiency improvement that by MSSA, and results that show EDAs can be improved by local search intuitively inferring this would also hold for higher-order units, there are no results that show what MSSA can solve that EDAs cannot.

Important distinction is that these algorithm decided the linkage and value of the variables.

Advantage - more efficient that EDAs on trap functions Cox and Watson (2014b); Mills et al. (2014); Iclanzan and Dumitrescu (2007), but no evidence of what can do that another cannot. In chapter **??** we show MSSA is able to outperform BOA even when the correct model is provided.

Main limitation is the model used is weak compared to EDAs, single level of organisation (even though called MSS)

The performance of MSSA is strongly influenced by the initialisation of a solution (the starting position in the solution space) and the neighbourhood of a solution).

The exist no neural network model that utilises the idea of multi-scale search - yet neural network models provide a method for multi-level representation, therefore providing a truely multi-scale search algorithm.

2.3.3.2 Local Search and EDAs

Substructure search Local search has been used to initialise a population of solutions to improve the performance of an EDA Pelikan and Goldberg (2006); Bosman and Thierens (2011). In doing so, it has been shown to significantly reduce the population size and also the number of function evaluations required for an EDA to find the global optimum solution Bosman and Thierens (2011). The performance improvement has been attributed to local search providing a clearer signal for the relationships between variables that improve the quality of a solution Radetic and Pelikan (2010).

Using local search to initialise the population of an EDA is often refereed to as a hybrid method. Hauschild and Pelikan (2011); Hsu and Yu (2015)

However, this hybridisation brings them more inline with the idea of multi-scale search

However, the use of local search is not consistent with the algorithm. Local search is generally only applied to initialise that solution and therefore seen as an additional process added to an EDA. Thus, EDA methods that use a local search to initialise a population are often referred to to as hybrid versions

Sub-structure search in EDAs Lima et al. (2006) exploit the linkage information contained within the probabilistic model. Applied via mutation operator Sastry and Goldberg (2004).

Due to the performance improvement provided by local search, it has been hypothesised that applying local search on newly generated candidate solutions would provide a performance benefit. - **memetic algorithms**. However, because the local-search is limited to the original representation, as the population search proceedes, the effectiveness of local search reduces.

The identification of the performance improvement local search provided to an EDA inspires the idea of incorporating local search in a more consistent way. Namely, by conserving relationships between variables, identified by the model to during local search. Lima Lima et al. (2006) exploited relationships in the Bayesian network of BOA to adapt the search operator for local search, searching in a substructure neighbourhood and subsequently improving the performance of BOA.

local search to initialise the population , significantly reduce the population size, and improve performance - clearer signal for building-blocks hybrid method, mutation operator

EDA's initialise using local search in the original variables. And know this improves performance. Therefore, it should be intuitive to abstract this to all scales of representation. Or at the very least, the model is capable of hill-climbing

As discussed later in this chapter, Model-building optimisation algorithm appear to dismiss the variation available by mutation. In fact, a single-point mutation applied to a EDAs is often refereed to as a hybrid algorithm - a mixing of local search and probabilistic generation. Yet, they are simply methods for providing variation.

2.3.3.3 Linakge Only Models

How these algorithms differentiate and relate to MMS and EDAs Where it currently is What is missing How this thesis relates to this

Similar to the idea of MSSA is the set of algorithms the linkage-tree Genetic Algorithm (LTGA) Thierens (2010), Parameter-less Population Pyramid (P3) Goldman and Punch (2014) and the Dependency Structyre Matrix Genetic Algorithm (DSMGA) Hsu and Yu (2015). The are separated from the MSSA, due mainly for what the model is used for. The model is used to only capture the linkage information between variables. The linkage information is then used as a crossover mask between two parent solutions. The algorithm is inline with MSSA becasue eat each

An alternative approach to MSSA is to construct a higher-order representation of the search operator Thierens (2010).

As highlighted in the review of multi-scale search algorithms, the models used are relatively simplistic in comparison to the state-of-the-art machine learning models and also in comparison to the SOTA EDA - hBOA. The algorithms appear suitably disjoint as to ignore one of the classes.

However, the Linkage Tree Genetic Algorithm (LTGA) Thierens (2010) bridges the gap between MSSA and EDAs. Further, LTGA outperforms hBOA on numerous benchmark problems in terms of scalability and computation time Goldman and Punch (2015); Thierens and Bosman (2013); Hsu and Yu (2015). As we review next, whilst not strictly and EDA, its similarity can not be simply ignored. Rather, it

More recently, the development of EDAs have moved towards using the model to inform partial changes to a solution by providing crossover masks. Approximating a method of searching in a new neighbourhood, but still no explicit

The separation building-blocks is referred to as linkage learning – which variables are members. Linkage learning doesn't associate with representing the partial solutions. It is only concerned with identifying which variables should be varied together. The value of these variables is not modelled. The values are taken from the populations

A model of the underlying problem structure is created by measure

Creating a model that represents which variables are correlated. Many methods concentrate only on linkage-learning. P3 and LTGA are state-of-the-art examples of this approach (literature recvew on these method - pelikan gave some).

LTGA Thierens and Bosman (2013) uses agglomerate 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 generated by applying a MIV method to the candidate solutions; referred to as 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.

The Linkage Tree Genetic Algorithm (LTGA) Thierens (2010); Thierens and Bosman (2013) uses an incremental tree linkage-set as the model. LTGA maintains a population of candidate solutions. The model is constructed using a distribution of solutions selected from the population using binary tournament selection. New solutions are generated by applying model-informed variation to all candidate solutions in the population. The method of model-informed variation is performed using optimal mixing: each solution exchanges information with all solutions in the population. The constructed linkage-set is used to determine which variables are exchanged between solutions. The process is repeated, constructing a new model each time.

The incremental-linkage set is generated by first constructing a dependency structure matrix (DSM). The DSM represents the pairwise dependencies between variables. The strength of the dependencies represents the measurement of mutual information between the two variables. That is, by observing one of the variables, how much information does this provide about the other variable. The matrix of pairwise dependencies, therefore, represents the partial-solution (building-block) structure, namely which variables should be exchanged together. Agglomerate hierarchical clustering is used to construct a hierarchical tree linkage-set. The clustering technique recursively combines clusters based on their dependency strength until only a single cluster remains. The nesting of clusters can only have a single parent and as such, creates a tree structure.

DSMGA-II The Dependency Structure Matrix Genetic Algorithm Version 2 (DSMGA-2) Hsu and Yu (2015) uses a incremental graph linkage-set as the model. DSMGA-2 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 applying model-informed variation to all candidate solutions in the population. The method of model-informed variation uses restricted-mixing and back-mixing. These methods are an extension of the ideas of optimal mixing. The methods of exchange information between solutions remain the same; the linkage-set is used to determine which variables exchange states between solutions. The subtle difference comes from deciding which solutions to uses for the exchange. The process is repeated, constructing a new model each time.

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 approximation maximum-weight connected sub-graph (AMWCS). The linkage set is, therefore, a graph structure: it is possible for a cluster to multiple parents.

P3

A type on incremental learning - online method that continuously updates the model - costly as many models need to be reconstructed at each iteration but very nice idea.

The parameter-less population pyramid (P3) Goldman and Punch (2014) 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 any state-of-the-art MBOA 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. Model-informed variation is then applied to the solution to improve its quality. If the solution is improved, it is now 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 model used by P3, as by LTGA, is a tree linkage-set that is constructed from a dependency structure matrix. Therefore, the types of relationships that can be modelled in P3 and LTGA are the same. The significant difference between LTGA and P3 is the use of multiple models. We hypothesis this enable P3 to solve dynamic optimisation problems. However, we do not explore this differentiation in this paper.

The improvement made to a single candidate solution in the population is determined by the diversity of alternative solutions available in the population and the genetic operators.

A type of multi-level representation - clearer in Chapter 4

2.3.4 Summary

There is significant overlap between MSS and EDAs, that is further explored in this thesis to distinguish there capabilities. It is hypothesised that the success off crossover, used in GAs is due to exploiting building-block structure. The extension to EDAs was aimed at explicitly building on this idea that identify and randomly recombining building-blocks is a good idea. MSS explicitly search's in building-block space using a hill-climber.

In all modelling approach's, it is evident that what makes a good model is that given all things equal, a simpler model. This is inline with generalisation in machine learning. Over-fitting the model will provide a good model of the training data, the solutions used to construct the model, however a poor performance for unobserved data points (solutions). Therefore using the model to perform recombination will be poor, it is desirable to have a model that can explore different areas of the search spcae that has not been visited. For this to be the case, the model must be a good model for these unobserved data points. The advantage of using a neural network model is that we can parameters this regularisation to improve the generalisation of the model.

Implicilty, EDAs and MSSA provide a mutli-scale representation by differentials in the strengths of their connections between individual units.

The Deep Optimisation algorithm combines multi-scale search with multi-level representation to explore the solution space.

2.4 Alternative Machine Learning Methods

Development of adaptive methods rely on identifying and exploiting the relationships between variables that guide the search to the global optimum. The assumption made by methods based on natural evolution is that the relationships form partial solutions, a sub-set of variable combinations that contribute to a solutions quality regardless of the assignment made to other variables. Therefore, the development of using automated methods for identify and capturing these relationships using machine learning methods is becoming increasingly more popular with application to many paradigms of search methods

Of course, in all this, learning is fundamental component to the success of optimisation algorithms. By learning and understanding the problem structure, we can design methods that's exploit regularities in the structure that enables search to find good quality solutions in feasible time. This thesis is interested in using learning methods to improve search in optimisation problems.

Alternative methods that use machine learning to improve optimisation include: learning a heuristic for a set of problem instances Zhang and Dietterich (2000); Khalil et al. (2017); Bello et al. (2016); using a surrogate model to approximate the fitness function Queipo et al. (2005); Vu et al. (2017); adapting the learning function to bias future search Hopfield and Tank (1985); Boyan and Moore (2000); embedding a machine learning model within the model of a combinatorial problem Lombardi et al. (2017) and using machine learning to select a suitable solver Volpato and Song (2019). The use of deep reinforcement learning algorithms for combinatorial optimisation is a popular approach Mazyavkina et al. (2020). 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 in general MBOAs, these methods don't use the model to recode the neighbourhood of a search space.

Machine learning used for other optimisation – branch and bound, learning heuristic – reinforcement learning. Learning which method to use – classification

2.5 Summary

Models in MSS provide a bottom up rescailing of changes made to a solution. In EDAs, the generation is top down constraint on the degrees of freedom. LTGA provides a multi-scale model of how to perform bottom-up or top-down MIV.

- NN models in an EDA framework doesn't show competitive results - Development has shifted away from EDAs and towards MSS - ltga, p3 - No algorithm has used a NN, let alone a deep NN, in the method of MSS - MSS a more natural fit for a deep representation - Before an in depth comparison between the state-of-the-art, we introduce DO.

As far as we are aware, there does not exist an algorithm that allows for exploring the interaction between developmental dynamics and population dynamics. In this thesis, DO provides a single an algorithm that is capable of being used in an EDA way and MSSA and thus providing a unification of the approach's and enables us to ask additional question related to the interactions between the two approaches.

Chapter 3

Searching in the Space of Building-Blocks

Deep optimisation is an algorithmic concept that expands the functionality of Multi-Scale Search Algorithms by using a machine learning model that has the capacity to representing deep features - features that are recursively constructed from lower-level features. The algorithm is based on the intuitive idea of taking a problem, decomposing the problem into smaller sub-problems, finding solutions to these sub-problems (features) and recombining the solutions to solve the original problem. As identified in Chapter 2. Recombination can be performed by random sampling (EDAs) or by local search in the space of solutions to the sub-problems (MSSA). Deep Optimisation will utilise deep learning to learn and uncover the underlying structure of the problem - providing and autonomous decomposition of the problem.

Overcoming ruggedness implied better operator and search space design. DO seems to fit nicely into this catergory – a hidden layer is learning a natural search space design that represents features present in the solutions (it is likely that these features contain good fitness benefits). Secondly, DO automatically designs its search operator, to that of searching in feature space. Mapping the solution space to a new space of features that should contain high-fitness contributions allows for both a more natural search space and search operator.

It is important to understand the characteristics of problems that deep optimization can solve and what other optimization algorithms, that are currently available, cannot.

This Chapter contains two components. The first stage is the understand the mechanism for using the model. Specifically, how search in the a space of partial-solutions differs (MSSA) from random combination of partial solutions (EDAs). Using a neural network within the functionality of an EDA has provided noncompetitive results. We therefore seek to understand how we can use the MSSA framework and a neural network. The rHN-G algorithm provides us with a foundation from which to develop deep optimisation. rHN-G uses a Hopfield network and Hebb's rule to learn associative relationships that are exploited to rescale the search operator. While the model capacity is weak in comparison to other models used, it never the less is successful at using the model in a new-way. In this Chapter, we first investigate how rHN-G can outperform BOA due to the way the information in the model is exploited. Using population sizing theory, Mills et al. (2014) showed that MSSA will find solutions non-additive modular problems in polynomial time, while EDAs will scale exponentially. We first expand on this work and confirm the result empirically. Then we perform a preliminary investigation using an autoencoder model to replace the Hopfield network used in rHN-G. We analyse the performance and find that we can successful use an autoencoder model to perform the same functionality as rHN-G by performing searching in the latent representation of a solution computed by an autoencoder model.. Further, we show that theoretical problems containing hierarchical problem structure can be efficiently solved using a single layered network, and thus conclude that their exists no theoretical problem in the literature that contains deep problem structure. Finally, we find that their does exist problem structure that rHN-G cannot represent and a neural network can, namely the 4-2-4 encoding problem . This provides sufficient preliminary results to further develop DO as expanded on in Chapter 4 and preliminary properties of theoretical problems required to demonstrate the performance of DO as expanded on in Chapter **??**

3.1 Filter and Constructive Selection

3.1.1 Introduction

In suitable domains, model-building optimisation methods can learn how to decompose an optimisation problem into nearly-separable sub-problems without *a priori* knowledge of the underlying problem structure. Identifying such modularity aims to exploit the familiar idea of separating a problem into smaller, simpler sub-problems, solving these sub-problems and then re-combing these solutions to solve the original problem. This forms the basis of the building block hypothesis where the combinations of alleles at strongly correlated variables in the solution space are conserved during future search, thus forming a lower dimensional structural representation of the original solution: a building block Holland et al. (1); Goldberg (1989). The idea is that building blocks are exchanged and combined, conserving the already-solved linkage structure at the lower level, to solve constraints imposed at the higher level (between modules). This allows the search process to be re-scaled from searching combination of discrete variables to combinations of building-blocks.

The central focus of this paper is to show that using selection during the generation of samples from the learnt model produces a significant efficiency gain over the more familiar use of selection which is to filter out candidate solutions from the pool of samples generated. To differentiate between these two uses of selection, we introduce the terms of filter selection (the conventional approach) and constructive selection:

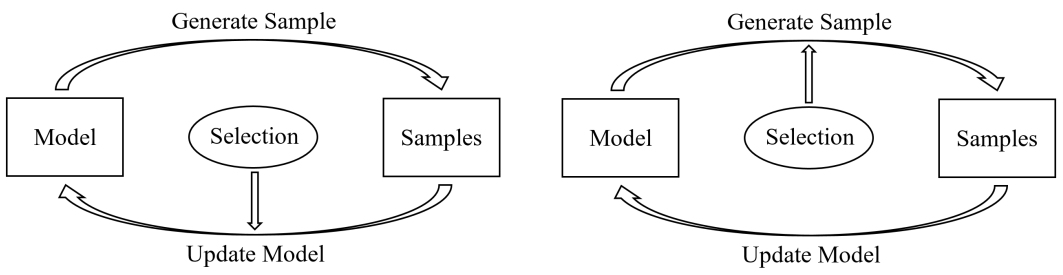
EDAs with Filter selection: The model is used to generate complete solutions without any input from selection. Selection is then used to determine whether that solution is good enough to be used to update the model. Fig. **??** illustrates this procedure.

MSSA wtih Constructive selection: The model and selection are used together to generate solutions. Specifically, selection is used (within an iterative process) to assess whether a model-informed modification to a solution improves its fitness. After generating the samples the model is updated accordingly. Fig. **??** illustrates this procedure.

Filter selection is the more familiar type of selection used in all genetic algorithms and Estimation of Distribution Algorithms (EDAs). These algorithms are stochastic optimisation methods which replace the GA's mutation and crossover approach by building explicit probabilistic models of fit individuals. They sample from this model to generate new candidate solutions, evolving a population towards the optimal solution Hauschild and Pelikan (2011). Concretely, the algorithms use the learnt model to bias the sampling of solutions. To assess the performance of filter selection we use the Bayesian Optimisation Algorithm (BOA) due to its ability to successfully solve a wide range of optimisation problems Pelikan et al. (2003); Aickelin et al. (2007); Santana et al. (2008).

BOA Pelikan et al. (1999) uses a sophisticated machine learning technique to automatically identify the underlying structure in the problem domain. Specifically, it models partial solutions probabilistically to identify a lower-dimensional model within the high-dimensional problem space. BOA is very capable of learning the structure of a problem given a population of samples containing a sufficiently strong signal of the underlying structure.

BOA works by initialising a population of candidate solutions from a uniform distribution of possible solution states. The fitness of individual solutions is evaluated. Truncation selection (this study used the top 50% of the population) is then applied on the population to filter-out promising solutions which are then used to construct the probabilistic model (a Bayesian network). From the model, new solutions are generated which replace the lowest fitness solutions in the population (this study replaced lowest 50% of the population). This process is repeated until the termination criterion is met. BOA is a clear example of using selection in what we term the 'filter selection' approach - i.e. to decide which samples are retained for updating the model and selection is not used during generation of new samples.



(a) Filter Selection: Selection pressure is applied to filter out a subset of solutions from the population

(b) Constructive Selection: Selection pressure is applied during the generation of a solution

FIGURE 3.1: Filter Selection applies to the group of solutions, an individual solution is updated if it improves the groups average fitness. Constructive selection applies a selection pressure directly to the individual solution. An individual is updated only if the individuals fitness is improved.

Constructive selection is an idea developed in recent work by Watson and colleagues in the development of multi-scale search algorithms (MSS). The idea has connections to new perspectives in biological evolution. Specifically, the notion of 'internal selection', a key idea in the extended evolutionary synthesis Pigliucci and Müller (2010); Laland et al. (2015), or 'developmental selection' Snell-Rood (2012) recognising the observation that the process of development is not a ballistic unfolding of genotype into phenotype but rather has elements of trial and error and feedback. For example, the wiring of synapses or the paths of blood vessels involves internal context-sensitive feedbacks that adapt to the details of the ongoing construction process as it happens. The action of selection is not simply to filter good phenotypes from bad, and the action of development is not simply to decode a genotype, but the two processes are much more intimately dialectic.

MSS algorithms are a class of evolutionary optimisation methods. Whilst the fundamental feature of repeatedly generating candidate solutions using a model remains, the generation of samples is adaptively and repeatedly redefined, using the model to create a new (lower dimensional) search space at successively higher levels of organisation. MACROMills et al. (2014), rHN-G Watson et al. (2011), Schema-grammar GA Cox and Watson (2014a) and BBHC Iclanzan and Dumitrescu (2007) are examples of this approach. Multi-scale search algorithms do not use a population of candidate solutions. Simply they generate partial solutions probabilistically from the model and use selection to determine if the partial solution is included into the complete sample. Whilst this is a type of EDA it uses constructive selection not filter selection.

In this paper, we evaluate the performance of constructive selection using the simplest of these methods - the restart Hopfield Network with Generative associations (rHN-G). rHN-G uses observations of its own optimisation dynamics to identify the problem structure. Its optimisation dynamics are essentially local search (hill-climbing) that is guided by selection, as normal, but also modified by biases provided by the (initially empty) model. The learned structure modifies its optimisation trajectories by redefining the variation operator - effectively exploring a lower dimensional representation of the solution space. It thus moves from searching the original high-dimensional problem domain to a lower-dimensional domain: transforming the dynamics to search combinations of discrete modules instead of the original problem variables. An overview of the algorithm is presented in section 3, for a more complete description refer to Watson et al. (2011).

We evaluate the algorithms' performance using an idealised nearly-decomposable constraint optimisation problem: the modular constraint (MC) problem Watson et al. (2011). Learning the structure of this problem is easy (the modularity is not deceptive), and if an algorithm exploits this structure correctly (i.e. enabling it to search combinations of modules) then the whole problem is easy. But if an algorithm cannot use the model to achieve this, then even though it might have identified the modules correctly, it will not be able to put them together to solve the whole problem Mills et al. (2014). This problem allows us to explore and exemplify the relative merits of constructive and filter selection.

This paper is not claiming that rHN-G is a better model building optimisation algorithm than BOA - BOA is a significantly more robust and general method than rHN-G. Rather, and more specifically, we aim to assess the value of the constructive selection approach in contrast to filter selection. Both algorithms can learn the modularity of our simple test problem successfully, but we show that the way in which the model is used, specifically its interaction with selection, enables rHN-G to get much more from the model than BOA does. We demonstrate this further by removing all the model building complexity from the algorithms (by giving them identical models of the modularity 'for free') and evaluating their ability to generate solutions from the model correctly.

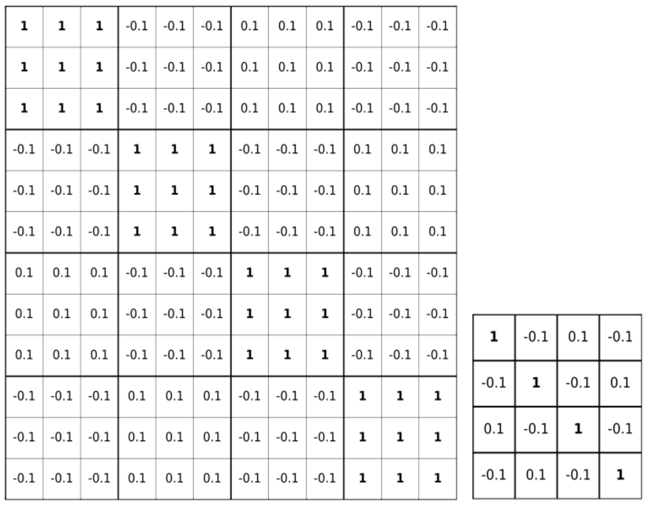
3.1.2 A Constraint Optimisation Problem with Simple Modularity

To evaluate the performance of the algorithms to solve modular constraint optimisation problems, specifically the scalability of their performance as a function of module size, we use the MC problem described by Watson *et el* Watson et al. (2011). This describes a simple 'nearly-separable' modular problem with one level of hierarchy Watson et al. (1998). The constraint problem is analogous to the natural energy minimisation of a dynamical system; equivalent to MAXSAT and resource allocation instances **?**.

The solution state to the problem is $S = \{S_1, \ldots, S_i, \ldots, S_N\}$, where $S_i \in \{-1, 1\}$ and N is the size of the problem. Here an energy function encodes the constraint optimisation problem by using a set of weights between all variables, S_i, that correspond to constraints imposed by the external environment. The fitness of a solution state is given by:

$$F_S = \sum_{ij}^{N} \omega_{ij} S_i S_j \tag{3.1}$$

where ω_{ij} corresponds to the constraint between variables i and j. Modularity can simply be incorporated into the constraint matrix by selecting strong weights for intra-module connections and weaker weights to describe inter-module connections. Fig 3.2 is an example of a constraint matrix imposed on the solution state for $N = 12$. It is a symmetric modular connectivity weight matrix with $n = 4$ modules each of size $k = 3$. One can see that by exploiting the problem structure encoded in the constraint matrix the dimensionality can be reduced to form a simple, easy to solve, random constraint (RC) problem as illustrated in Fig.3.2. However, without exploiting the modular structure the MC problem is very difficult to solve. Given that the inter-module constraints are small in contrast to the intra-module constraints creates local optima that correspond to $S = \{S^k, -S^k\}^n$ For example, the state configuration $S = (1^3, 1^3, 1^3, 1^3)$ is a local optimum on the fitness landscape for the MC problem in Fig. 3.2. The global optima would be $S = (1^3, -1^3, 1^3, -1^3)$ or its compliment.



1	1	1	-0.1	-0.1	-0.1	0.1	0.1	0.1	-0.1	-0.1	-0.1
1	1	1	-0.1	-0.1	-0.1	0.1	0.1	0.1	-0.1	-0.1	-0.1
1	1	1	-0.1	-0.1	-0.1	0.1	0.1	0.1	-0.1	-0.1	-0.1
-0.1	-0.1	-0.1	1	1	1	-0.1	-0.1	-0.1	0.1	0.1	0.1
-0.1	-0.1	-0.1	1	1	1	-0.1	-0.1	-0.1	0.1	0.1	0.1
-0.1	-0.1	-0.1	1	1	1	-0.1	-0.1	-0.1	0.1	0.1	0.1
0.1	0.1	0.1	-0.1	-0.1	-0.1	1	1	1	-0.1	-0.1	-0.1
0.1	0.1	0.1	-0.1	-0.1	-0.1	1	1	1	-0.1	-0.1	-0.1
0.1	0.1	0.1	-0.1	-0.1	-0.1	1	1	1	-0.1	-0.1	-0.1
-0.1	-0.1	-0.1	0.1	0.1	0.1	-0.1	-0.1	-0.1	1	1	1
-0.1	-0.1	-0.1	0.1	0.1	0.1	-0.1	-0.1	-0.1	1	1	1
-0.1	-0.1	-0.1	0.1	0.1	0.1	-0.1	-0.1	-0.1	1	1	1

1	-0.1	0.1	-0.1
-0.1	1	-0.1	0.1
0.1	-0.1	1	-0.1
-0.1	0.1	-0.1	1

FIGURE 3.2: A small example of the modular constraint (MC) problem (left) and equivalent macro-scale representation random constraint problem (right).

Fig. 3.3 shows the fitness landscape that corresponds to an MC (problem of $N = 100$), where the extremes of the x-axis denote the global optima. A local search algorithm that cannot exploit the problem structure will get trapped in the local optima. In the attempt to satisfy an inter-module constraint, it will cause a reduction in the fitness due to violating the larger intra-module constraints imposed on the state variable.

With this problem definition, we can create an idealised nearly-decomposable constraint optimisation problem which has clear structure that can be controlled. This allows us to draw direct comparison between the algorithms and focus on evaluating the performance and sensitivities to modularity

For the experiment, we use a MC problem where the number of modules remains constant ($n = 10$) and only the size of the module (k) is varied ($N = 10k$). The results and evaluated time complexity required for the algorithm to solve the problem are therefore only in reference to the size of the modules. For all problems, the intra-module constraint strengths are set to 1 and the inter-module constraint strength (p) are set to $1x10^-6$ (this is a known value that creates an MC problem that is pathologically difficult for a hill-climber Mills (2010). The inter-module weights between different modules vary only by their sign, acting to attract (same sign) or repel (opposite sign) each other. These are specifically chosen to produce a global optimum with alternating module solutions, i.e. $S = (1^k, -1^k, 1^k, -1^k, 1^k, -1^k, 1^k, -1^k, 1^k, -1^k)$ or its compliment. The extreme imbalance between intra-module and inter-module weights means that the basins of attraction for each local optimum in the high-dimensional problem are almost equal; revealing almost no inter-module correlations. A single run with a hill-climber will be no more likely to find the global optimum than any other local optimum. However, in principle, an algorithm that is able to learn and properly exploit knowledge of the modular structure has potential to 'hill-climb' in the space of module combinations, following the relatively weak inter-module fitness gradients to locate a global optimum.

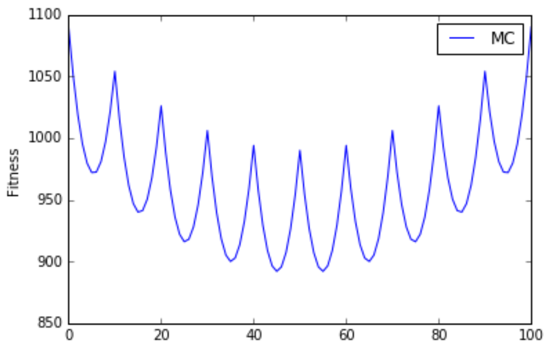


FIGURE 3.3: A small example of the fitness landscape cross-section for an MC problem, with $n = 10$, $k = 10$ and $p = 0.01$. In this illustrative example, we can see that intra-module constraints create many local optima and inter-module constraints create a relatively weak gradient at a macro scale. This macro scale gradient can be used to hop from one local optimum to another, and hence find the global optimum, if and only if the algorithm can utilise its model on the modular structure appropriately.

3.1.3 Methods

Learning and exploiting the modular structure of the problem domain is fundamental to solving the MC problem. Whilst there exist sophisticated algorithms for learning the problem structure without prior knowledge, we argue, that by using selection within the process that generates samples from the model it is possible to exploit the learnt information in the model in a more efficient manner.

3.1.3.1 BOA Parameters

Here the threshold termination criterion of 95% is used to determine when BOA stops the iterative process. Assuming that a global optimum was found, this termination criterion means that using the model and sampling from it would produce a global optimum 95%

of the time. Alternatively, from an evolutionary view point, the population contains at least 95% of globally optimal solutions. To determine the time-complexity for BOA in relation to the module size only the number of fitness evaluations are counted to reach the termination criterion (i.e. model induction is assumed to be 'free' for the purposes of this study). A pre-experiment study was used to find the minimum population size for BOA that finds the global optimum at least 50% of the time. This provides a 'generous' measure of the minimum number of fitness evaluations required for BOA to solve the problem. Only the successful runs of BOA are used to formulate the distribution of fitness evaluations required to solve the problem (hence consisting of 50 samples for each test data point).

3.1.3.2 rHN-G

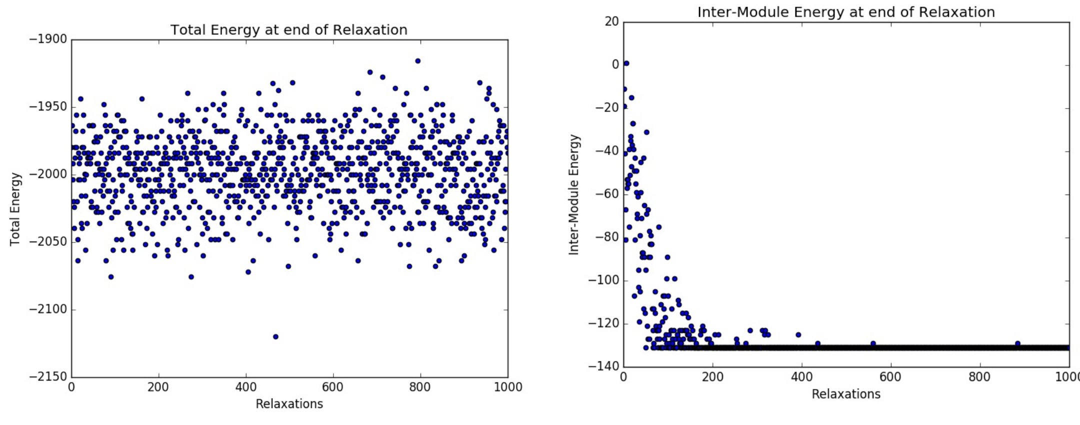
By restarting the candidate solution from a uniform random distribution enables to search

The time-complexity of rHN-G in relation to the module size is measured as the number of fitness evaluations required to (build a model that can) reliably produce the global optimum. Reliably is defined here as being able to produce the global optima at least 95% of the time when given a random solution string. As with BOA, 50 repeats are performed to produce a distribution of the total number of fitness function evaluations required to solve the MC problem. This is a product of the number of iterations for each constructive selection process and the number of solution restarts performed (training examples used). The number of steps (iterations) per a constructive selection process was not optimised here, and instead $10N$ is used as done by Watson et al. (2011). All the fitness evaluations used within the generation of the sample are counted in the fitness evaluations reported. Furthermore, the learning rate was not optimised for different k values and was kept constant at 0.0002. The optimisation of these parameters was not necessary - although this would reduce the total number of fitness evaluations, the efficiency gain by using constructive selection would still be dominant.

As rHN-G updates a single solution during optimisation, it allows us to measure the effect of the model on optimisation directly. Where as in BOA, the effect of the model is at the population level.

At the initial stages of rHN-G, when the model has negligible information about the problem structure, the constraints between the bu idling-blocks are ignored in favour of satisfying the higher-rewarding intra-module constraints, Figure: Figure **??**. Indeed, when no learning is applied we see that rHN-G is provides a distribution of local optimal solutions, but is unable able to find the global optimum solution.

- How does rHN-G work - change in the weights as optimisation is performed.



(a) Energy state of solutions found using only single-bit substitution (no learning)

(b) Trajectory of 10 independent solutions that use only a single-bit substitution.

FIGURE 3.4: Energy state of solutions found using with learning

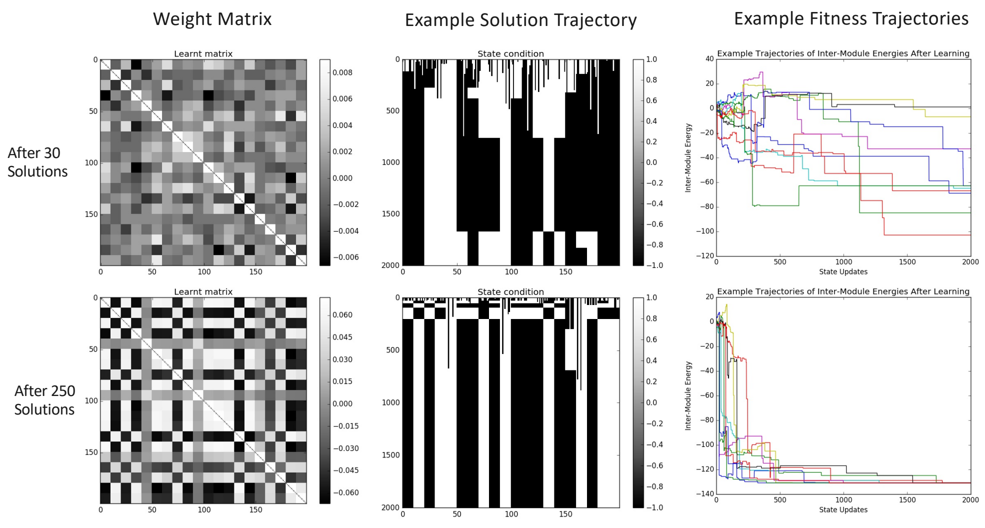


FIGURE 3.5: Average number of fitness evaluations required to solve the MC problem, size $N = 10k$. The error-bar shows the range of values from the test distribution. The solid lines are trend lines calculated from the average data points, BOA results use an exponential fit and the rHN-G result uses a linear fit.

3.1.4 Results and Discussion

Experiments to calculate the number of fitness evaluations required to solve the problem for different size modules is conducted. For BOA, we also present results using various model complexities for the Bayesian network. This is to further highlight that the difference between the algorithm's performance is in how selection is used when generating candidate solutions. The complexity is controlled by parameter MaxIn which limits the maximum number of incoming edges into a node of the Bayesian network.

Fig. 3.6 presents the average number of fitness evaluations required to solve the MC

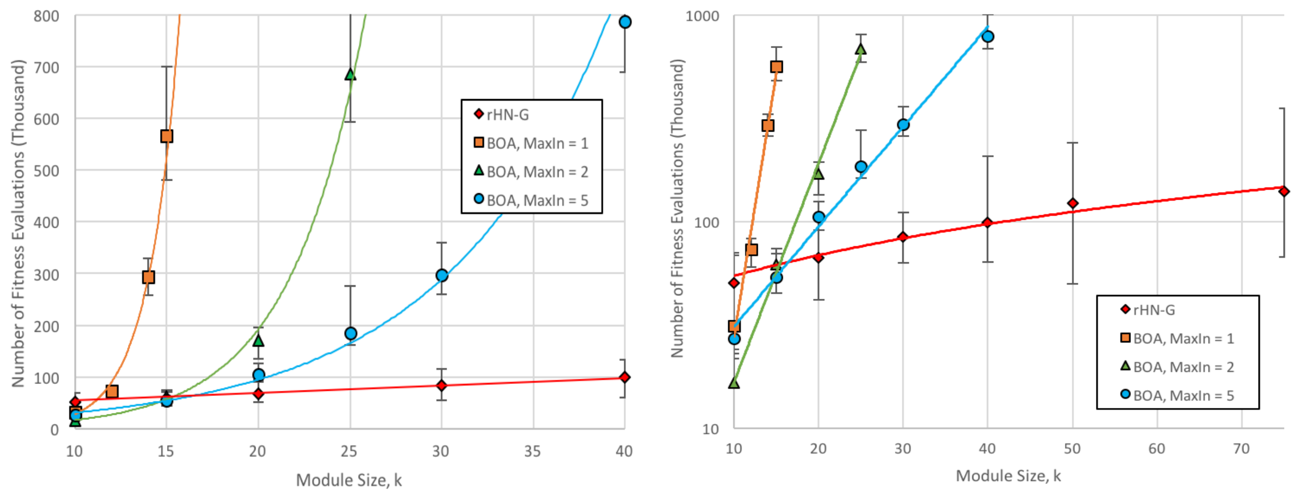


Figure 3.6: Average number of fitness evaluations required to solve the MC problem, size $N = 10k$. The error-bar shows the range of values from the test distribution. The solid lines are trend lines calculated from the average data points, BOA results use an exponential fit and the rHN-G result uses a linear fit.

problem. The error bars represent the maximum and minimum number of fitness evaluations observed in the 50 independent repeats. To provide clarity on the performance of rHN-G the results are also plotted using a log-scale for the y-axis in the same figure. The superior performance of the rHN-G method allowed us to quantify results for significantly larger problem sizes than for BOA. The solid lines are trend lines. For all the BOA experiments the number of fitness evaluations scales exponentially with module size. Even when the complexity of the model increases significantly beyond the minimum necessary for this problem, the time-complexity remains exponential in module size. It is evident that rHN-G is significantly more efficient here. It is not scaling exponentially; a linear, logarithmic and fractional power (0.5) is satisfiable given the error margins. Presented in the figures is the linear trend line as this best follows the trend of the data points.

Investigations show that for the smaller module problems the number of fitness evaluations necessary for rHN-G to converge can be reduced by increasing the learning rate. Investigating the exact dependency of the learning rate in relation to the module size does not affect the conclusion of this paper and is not investigated here.

The computational time-complexity to generate the Bayesian network is not included in our analysis (only fitness evaluations). Thus, the efficiency gain seen in BOA by increasing MaxIn is misleading. Specifically, the time-complexity for constructing the Bayesian network scales exponentially with the maximum number of incoming edges ($MaxIn$) to a node in the Bayesian network as $(N_p)^{MaxIn}$ where N_p is the number of problem variables. Although this is a significant computational cost, here we are only interested in the efficiency gained by the different ways of using selection in generating samples and not in the complexity of building the model. The results show that by using constructive selection in replacement of filter selection provides a significantly more efficient algorithm. As such it is possible to use a simplistic model building technique to solve a modular constraint problem that BOA finds difficult.

In the following subsection we separate the components of the time complexity that derive from model induction and from generating samples. But first we simply compare the models head-to-head including all fitness evaluations for model induction and generating samples (BOA does not use any fitness evaluations in the latter because it does not use constructive selection). In this comparison, module size is the key parameter of problem difficulty.

Widely used in evaluating the performance of model-building optimisation methods is the deceptive-trap (Trap-K) optimisation problem Deb and Goldberg (1993). For this problem, local optima are created in modules/blocks such that the local fitness gradient would lead the solution away from the global optima. Fit solutions, specifically local optima, would therefore not contain information about the global optima. The algorithms therefore require a population size sufficiently large to already contain the solution to the building blocks. This requirement scales exponentially with the size of the module. Ingrained in the literature is the notion that modules/blocks had to be deceptive in-order for the problem to be unsolvable for a simple hill-climbing algorithm (therefore requiring an EDA). However, the module size had to be small due to the exponentially scaling. If a module was not deceptive then the size of the module could be large. However, the problem becomes simple such that a hill-climbing algorithm can solve the problem. This is a false dichotomy. Mills Mills (2010) showed that an optimisation problem with an obvious nearly-separable modular structure can be pathologically difficult to solve for a hill-climber, but still easy to solve for a model-building method despite having large modules. For such a problem, local search is able to solve any one building-block semi-reliably (there is no need for exponential search to find the intra-module solution). However, a simple hill-climbing algorithm would fail to solve the entire problem as solving each module optimally is not fully reliable (see cross section in Fig. 3.3). Nonetheless, a sample of local optima is sufficient to identify the modular structure, and if used appropriately, this structure can be used to find a global optimum by guiding the substitution of one module-solution for another.

For the deceptive-trap problem it is acknowledged that the scalability of model induction is exponential in the module size. The result here confirms that even for a modular constraint problem where the solutions to the blocks can be easily found (blocks are not deceptive), BOA still scales exponentially in relation to the size of the module. This confirms the theoretical analysis performed by Mills *et al* Mills et al. (2014) which showed an algorithm like BOA would scale exponentially whilst a multi-scale search algorithm would scale polynomial with respect to the size of the module. Thus, the exponential scaling here isn't a consequence of the deception; BOA requires exponential time to solve constraint optimisation problems with large modules, deceptive or not.

3.1.4.1 Efficiency Gained Using Constructive Selection

Here we examine exactly the difference that constructive selection is having on these results by using a control experiment that removes all the differences due to model induction.

BOA and rHN-G have very different methods of identifying and learning the structure of the problem domain. Whilst there is an obvious disparity between the sophistication of the model building techniques, namely induction of Bayesian network compared to Hebbian learning of a simple pairwise correlation matrix, it is important to highlight exactly what is being learnt by the models. BOA will learn from correlations that have been filtered to have above-average fitness. rHN-G on the other hand, learns from locally optimal solutions. This has the effect of increasing the 'signal' of high fitness correlations from the 'noise' of spurious ones. It is therefore possible to argue that the difference observed in the experiments above is due to the difficulty in learning the model as one is using fitter training examples than the other. It is acknowledged that BOA requires a large population in order to learn large modules and the model used is more sophisticated than necessary to learn the problem structure we define here (a pairwise correlation model is sufficient). All of which would suggest that rHN-G would be a faster algorithm than BOA for this problem thus confounding our main aim in this paper, i.e. the assessment of constructive selection. To remove all these issues, we remove the complexity of the model building from the algorithms to specifically show the time-complexity difference between filter selection and constructive selection. In reference to Fig. **??** and Fig. **??**, for both methods we are simply removing the time complexity of the 'Update Model' arc from the procedure, thus only evaluating the performance of selection and generation (the 'Generate Samples' arc).

For both methods, we are providing models that identify and solve the modules but we do not provide information on how to put the modules together to find the global optimum. Learning the inter-module dependencies from a sample of local optima is very difficult for both methods because good combinations of modules occur only very slightly more often than incorrect combinations of modules in the MC problem. But for rHN-G this is not an issue. rHN-G does not need to learn the inter-module connections to solve the problem: the model of the intra-module connections plus constructive selection puts the modules together correctly and easily. In this control experiment, both BOA and rHN-G are restricted so that no model learning takes place during their iterative search, thus only comparing how the solutions are generated and selected to solve the optimisation problem.

Fig. 3.7 shows an example of the model given to BOA (for $n = 10$, $k = 10$) showing the edge connections between variables in the Bayesian network. It has been designed such that the first variable in the module is the root node of each sub-network. All subsequent variables within this module are connected to this root of the module and the probability

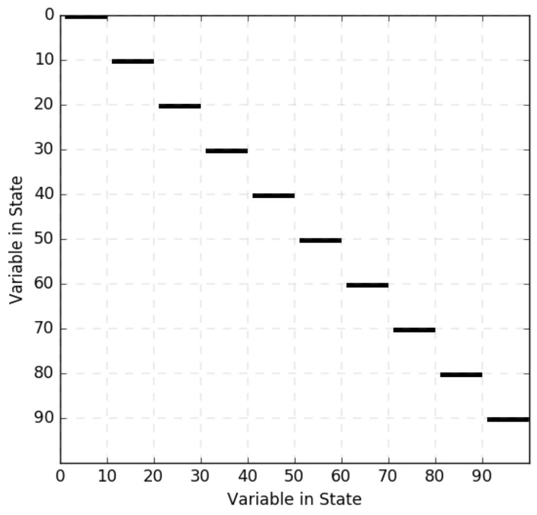


Figure 3.7: The Bayesian network connection matrix when BOA is given the correct model to identify the modular structure of the problem for a MC problem with N=100, n=10.

of having the same value as the root is equal to 1. There are no connections between modules and no learning of the inter-module structure can take place (the probability of the root node having value=1 is 0.5). This model correctly identifies all intra-module connections and for each generation of a solution, the intra-module constraints will be satisfied. Giving BOA the correct model from the outset shows that the algorithm is not miss-identifying the model. The same is also done for rHN-G. The exact correlation matrix for intra-module connections is given: the leading block-diagonals in the model (M) are equal to 1. As such each iteration of the constructive selection process will perform only a single block substitution. Therefore, for both algorithms, the intra-module constraints are solved and the models are given such that the search has been re-scaled to a lower dimension: the algorithms need to only now satisfy the inter-module constraint problem.

One can see that providing the model to the algorithms removes the dependency on the size of the modules: the only remaining problem to solve is the inter-module constraints. There is therefore no remaining sensitivity to k (module size), but a dependency on the number of modules in a problem is still present. For this experiment, we therefore keep the size of the module constant ($k = 10$) and vary the number of modules in a problem. The algorithm is terminated when it has first found a global optimum. 50 independent runs for each algorithm were performed to produce an average of the number of fitness evaluations required to find a global optimum.

Fig. 3.8 presents the results for BOA and rHN-G given the model. It is clear from this figure that even given the correct model of the modular structure, BOA is still unable to solve the problem in polynomial time. As before, the solid lines are trend lines. BOA required an exponential fit, specifically $0.5(2^n)$, where as the rHN-G only required $1.1(n \log n)$. This result confirms that the efficiency saving seen between BOA

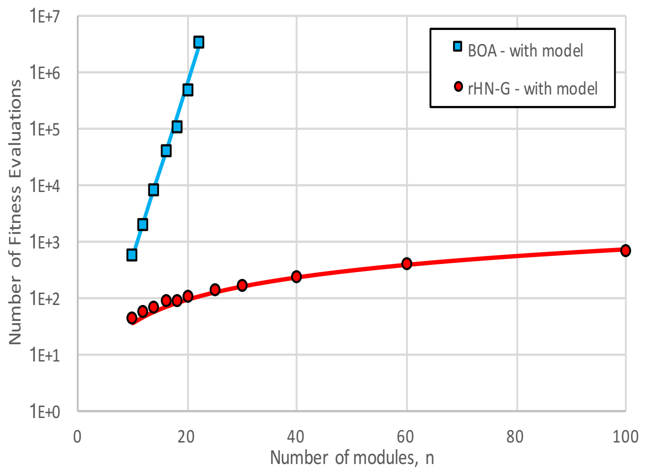


FIGURE 3.8: Average number of fitness evaluations required to solve the MC problem given BOA and rHN-G have been supplied with the problem structure. BOA scales exponentially (2^n) and rHN-G scales as $n \log n$.

and rHN-G is not a result of the difference between the model induction complexities. This experiment removes any difficulty with identifying and learning the structure. It is rather the difference between using filter selection and constructive selection.

The results from this experiment suggests that the efficiency of model building optimisation methods can be increased significantly by exploiting the information that they learn differently. In simple modular problems like the one examined here, this saving is due to the fact that a method using only filter selection cannot generate optimal solutions until it has learnt the dependencies between modules (as well as within). Constructive selection does not need to do this; constructive selection can search combinations of modules (using the model) and incrementally improve these combinations via iterative feedback with selection.

It would be possible to incorporate a constructive selection method into other EDAs, enabling access to the advanced model-space of BOA, for example, and more efficient use of the model that is learnt. We also note that rHN-G uses only constructive selection, but filter selection and constructive selection are not mutually exclusive in general. Their combination remains for future work.

We note that another way for the model and selection to work together to create solutions is utilised in the Linkage Tree GA Thierens (2010). The approach has enjoyed considerable success with the use of model-informed crossover - which like constructive selection - uses the identification of problem structure to effect module substitutions. We think that, at some deep level, the use of constructive selection and the MSS approach is conceptually similar to model-informed crossover, but whereas the LTGA uses the model to determine how to combine two solutions (during crossover), constructive selection uses the model to adapt a single candidate solution. Assessing the significance of this distinction remains also for future work.

It is hypothesis that LTGA will also be successful, like rHN-G for this type of problem as it also re-scales the crossover operator and performs updates similar to a hill-climber. Indeed, as we see in Chapter **??**, where a comprehensive comparison between the SOTA methods is performed, LTGA performs similar to rHN-G and DO on these types of problems.

Whilst the focus of this paper has been exclusively on computational efficiency, the implications for our understanding of biological evolutionary processes is significant. Specifically, the intuition that internal selection is important in biological evolution Laland et al. (2014) is supported by these results - suggesting that some adaptations (e.g. simple kinds of modular phenotypes) cannot be evolved successfully without constructive selection but can be evolved easily with constructive selection.

3.1.5 Conclusion

This paper has introduced the classification of constructive selection to differentiate from the conventional (filter) use of selection. We have found that by using construction selection in model-building optimisation methods results in a reduction in time-complexity required to solve a modular constraint optimisation problem from exponential, using filter selection, to polynomial, using constructive selection. We verify that this result is caused by differences in the ability to generate good samples from the model and not from differences in model induction complexity. We show this by using a control experiment that removes all differences in model induction by giving both methods an accurate model of the modules in the problem. The constructive selection approach can be incorporated into other more-sophisticated EDAs to boost their ability to exploit learnt models in the same way. This suggests that by using constructive selection other EDAs could solve problems that they are otherwise unable to solve in polynomial time.

BOA needs to learn how to combine the modules together. If it looses some of this information, the algorithm will fail. rHN-G on the other hand, and in more general MSSA, do not need to learn how the modules should combine. The algorithm is able to simply follow the gradient to find the correct combination. We revisit this idea when we compare the SOTA algorithms where for problem in learning the immediate single for how modules combine causes failure, where as searching in the space and only learning the structure at the basin of attraction solves the problem easily.

3.2 Multi-scale Search Using a Neural Network

A deep neural network provides a model space that can explicitly represent multiple scales of variability in a data.

Multiscale search – rhng – rescaling variation but limited by the model (correlation matrix)

The rHN-G algorithm was capable of efficiently solving the modular-constraint optimisation problem. The performance is attributed to the ability of the model to correctly identify and capture the building-block structure in the problem and then exploit this information by making partial updates to a solution (rescailing the search operator to higher-orders of organisation), where selection immediately informs the update before any replacement is made. In the case of BOA, replacement occurs before selection. Further, even when the model has the precise information about the building-block structure, the method for exploiting this information is not able to find the correct combination - due to the random sampling method. However, a weakness of rHN-G is the model used to capture the problem structure - a correlation matrix. Considering the sophistication of modelling approaches available and the wide success, especially deep learning, it is hypothesized that by exploiting the success of deep learning to learn complex functions and generalize to all types of tasks will enable a MSSA to solve complex and large instance optimization tasks. The Bayesian network model used by BOA is far superior and capable of capturing more complex and interesting structures that rHN-G. In addition, rHN-G is classed as a Multi-Scale search algorithm, where it recursively rescales the variation operator to higher-orders of organisation. However, the model use does not explicitly represent multiple-levels of representation. We therefore hypothesis a model capable of explicitly representing multiple-layers of representation, such as a deep neural network model, will be capable of representing problem structure that rHN-G cannot.

In this section, preliminary experiments are performed to understand how a neural network model can be used to provide the same functionality as rHN-G. Specifically, searching in the representation of building-blocks. Next we explore multi-level problem structure using the Hierarchical If and Only If problem (HIFF) Watson et al. (1998) to understand how the algorithms, and in particular the models overcome the challenge of hierarchical problem structure. We find that both rHN-G and a single layer neural network model is sufficient to solve HIFF, yet the models used are shallow. We therefore conclude that HIFF is not a deep problem and can be sufficiently approximated using a shallow model. Finally, we use inspiration from the theory of machine learning to identify building-blocks that cannot be represented by pairwise associations. Experiments show that the Hopfield network is unable to capture the relationships of the building-block where as the neural network model can. This provides preliminary evidence that the neural network is capable of learning and exploiting problem structure that other models cannot. As such, this provides sufficient direction to develop Deep Optimisation, as performed in Chapter Chapter:DOAlgorithm

3.2.1 Restart-Autoencoder with Generative Associations (rA-G)

The autoencoder is trained, using propagation, to output the input. The reconstruction error is used to determine the adjustment to make to the network weights. The adjustment will amplify some connections and remove others. In a ideal situation, neurons will respond to salient features that are independent at one level of organisation, and deep layers will represent relationships between these features (capture the interaction between the lower-level features).

An intuitive example of why deep networks are useful is presented in Figure Figure 3.9. Here the first layer neurons represent edges in the images. The next layer combines these edges to make a more complex features such as eyes and nose where the third layer now combines these to make faces. This property is what precisely what is desired. In engineering and optimization tasks the whole problem is separated into small sub-problems (the edges) combining these sub-problems to for higher-dimension representations (eyes, noses).

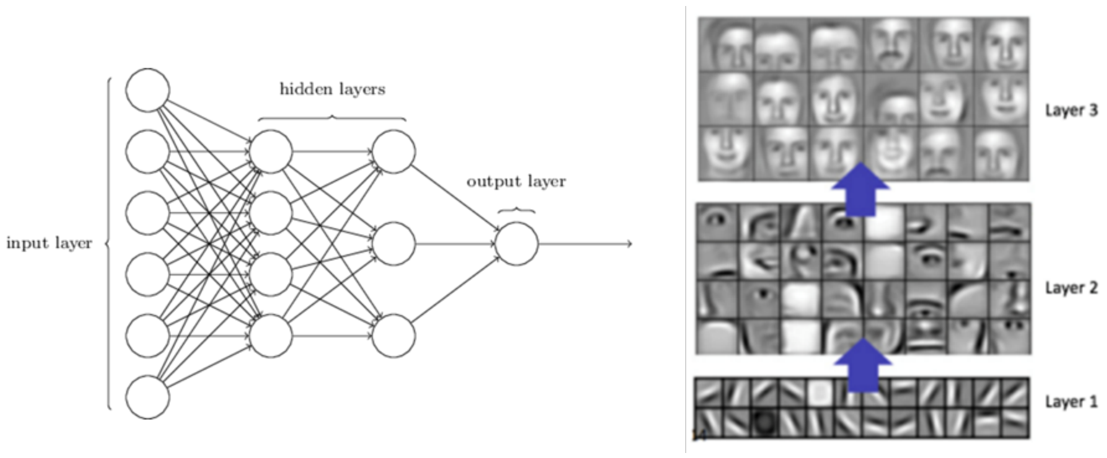


FIGURE 3.9: Example of a deep neural network extracting and representing higher-order features at the hidden layers.

Autoencoder Model

An architecture that fits naturally with creating high-levels of organization (forming building blocks) is an auto-encoder deep learning model.

By applying constraints on the network, such as having a lower number of neurons on the hidden layer in comparison to the number of input neurons the auto-encoder is able to learn and uncover interesting structure from the data.

This is a simple archetechure that is hyporthesised to produce the behavior desired whilst also allowing clear investigation into what it has learnt. A compression of the modular constraint problem should directly give the building blocks, which are clear. As such the neurons in the hidden layer should be responding to these building blocks. Before implementing this into the genetic algorithm which uses constructive selection, experiments are firstly conducted to verify that it can indeed learning the structure to the modular constraint problem and that the information is available such that it is possible to extract and use in a constructive selection way.

The encoder and decoder network are updated during training. Only the decoder network is used for generating reconstructions from the hidden layer. DO uses an online learning approach where the learning rate controls the ratio between the exploration and exploitation of the search space.

- Autoenocder a simple, well studied model that has a dimensional reduction interpretation. We perform our preliminary experiments using this model archetecture. It uses an encoder and decoder model. The encoder model transform the data into a higher-level representation (the hidden nodes) referred to as the latent space. The decoder network transforms the latent representation back to the original input representation. A good model is capable of producing a good reconstruction of the input from the latent representation. - neural network use hidden nodes to represent features from a training set - deep networks construct successively higher-order and abstract features to separate the classes. - Single neuron is an activation function of the weighted sum of its inputs. The activation function is non-linear allowing for multiple layers - Architecture is set 'a prior' and trained using back-propagation which moves the weights in the direction that reduces the error. The error is calculated as the difference between the desired output and the output of the network given the input. - For an autoencoder, the desired output is the same as the input. Therefore, the autoenocder is trying to learn the identity function and is a type of unsupervised learning algorithm as the data is not require labelling. A bottleneck is placed at the hidden layer to encourage the model not to learn a trivial identity function, but to instead learn a dimensional compression of the data, where the hidden neurons represent the most salient of features from the data-set.

Algorithm of rA-G is presented in Algorithm 3

The auto-encoder is trained in an online method

rA-G updates a solution using model informed variations.

During the development of using the autoencoder within the algorithm it was very important that the optimization algorithm was able to still perform bit-flips (varational operations at the lowest level). This is achieved by randomly selecting which level to use for generation, the input level or hidden neuron level. If the input level is selected then a simple bit-flip is performed. At initialization, the neural network will contain no information thus informed variation from the model will not update a solution and variation rejected by selection. It is therefore important that this extrapolates to future deep models, it needs to be able to perform variation at all levels of organisation. For here, we simply rely on random selection of a layer.

Evaluation of rA-G on MC problem

Algorithm 3: rA-G Algorithm

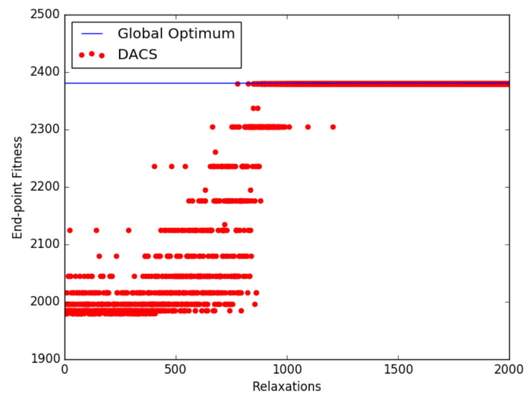
```
Initialize: Model;
while Termination Criteria not met do
    Initialize: Solution using random uniform distribution;
    while Optimising Solution do
        Randomly Select a Representation Layer if Representation Layer = Solution
          Layer then
            Make a single-bit substitution to the Solution representation ;
        else
            Use encoder network to calculate a latent representation of Solution;
            Make a single-bit substitution to the hidden representation ;
            Use decoder network to transform the latent change to the Solution level
              and apply to Solution ;
        if Change to Solution improves Solution quality then
            Accept change to Solution;
        else
            Reject change to Solution;
        Train the Autoencoder model using the optimised solution;
```

The performance of rA-G is evaluated using the MC problem containing 200 variables (N=200) with 20 building blocks (n=20) of size 10 (k=10). The inter-module constraints were chosen as to produce a global optimal solution with alternating solutions to the blocks. S = (1k, -1k , .., 1k , -1k). Here there are 2^{20} ways to put the modules together, and therefore an impossible task for a hill-climber or random search to achieve. The number of solution optimisation steps was set to 2000. After each stage of solution optimisation, the model the autoencoder model is updated.

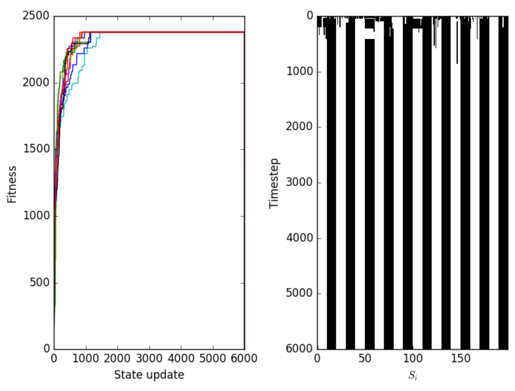
40 hidden neurons were used in the hidden layer (fully connected and only one layer) – more than sufficient to represent the structure (building-blocks) within the problem. a learning rate of 0.01 was used. Online learning was used, specifically the weights in the autoencoder applied one back-propagation step to the neural network using the developed candidate solution. The weights are initialized using a random uniform distribution between 0.1 and -0.1.

To confirm that by learning the problem structure makes the problem easy to solve, and verify that rA-G is producing a distribution of locally optimal solutions. rA-G is run with a learning rate of 0.

As often done by machine learning researchers to gain an understanding for what the model has learnt, we can visualise the weights of the layer. As the problem structure is well-defined it is possible to interpret the weights as a function result of learning the problem structure. Figure 3.14 presents the evolution of the neural network weights as it learns from more solutions. As the algorithm progresses, a building-block appears as a set of strong connections between a hidden neuron and each variable in the building-block.



(a) Energy of solutions found using rA-G



(b) Trajectory of 10 independent solutions that use only a single-bit substitution.

FIGURE 3.10: Energy state of solutions found by rA-G

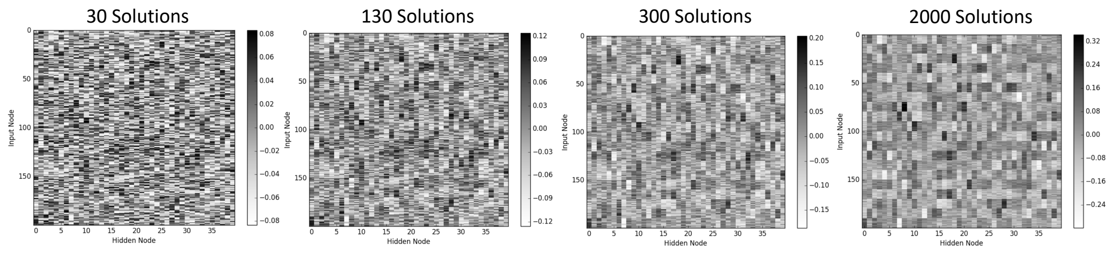


FIGURE 3.11: The weights in the neural network learn the building-block structure.

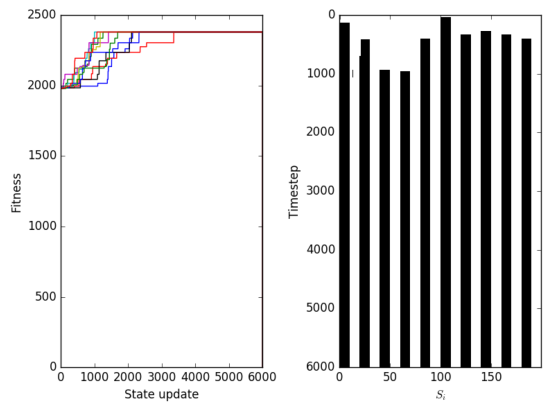


FIGURE 3.12: Given a locally optimal solution, furthest from the global optimum, rA-G is capable of escaping the local to find the global optimum solution.

As we see from this figure, the representation is not clear considering the simplicity of the problem. A building-block is represented by many hidden neurons.

rA-G is able to escape the local optimum that is furthest away from the global optimum, requiring a total of 10 module flips. Figure Figure 3.12 ACS is capable of finding the global optimum with the learnt model.

This experiment has provided two interesting insights into the behavior of the autoencoder for learning the structure for the MC problem. Firstly, the neural network is able to correctly identify the locations of the building blocks present in the problem. Secondly, it is able to also contain information on how some of the building blocks should go together. Crucially this is achieved by just using a single layer. This is because of using the tanh activation. If a sigmoid activation was to be used, it would be expected that the first layer would represent just single building blocks, or a neuron would only represent building blocks that have the same value, the second layer would then represent how to combine the building blocks. Therefore, at this stage, it is advantageous to use the tanh activation function as a single layer neural network is sufficient. Training of deep networks adds further complexity to the learning. The use of the tanh function is useful as it represents the transformation of the building blocks: this makes it simple for development when attempting to construct building blocks from the autoencoder. Tanh also has the advantage of propagating errors backwards.

The experiment has clearly demonstrated that the autoencoder can sufficiently decompose the problem into sub-problems, the information required to do so is present in the magnitude and the values of the weights. The next experiment is attempting to extract the information from these weights, such that it is possible to perform a partial update to the current candidate solution using the constructive selection approach.

3.2.1.1 Hierarchical Problem Structure

We have seen that a neural network model is capable of identifying and capturing building-block structure and exploit this information in the same mode as rHN-G. An important distinction between rA-G and rHN-G is that the autoencoder model can explicitly represent a deep structure by stacking multiple hidden layers on top of each other. This raises the question, what problem structure can a deep network learn and represent that a shallow network cannot. Our preliminary investigation starts with understanding the performance of rHN-G and rA-G on a hierarchical optimisation problem.

For this we use a hierarchical version of the MC problem. The hierarchy is created by seperating the strengths of the connections between variables

HIFF Problem - Definition

rHN-G can represent hierarchical problem structure when a tree As building-blocks are formed in a probabilistic way, it allows for exploitation of multiple levels of representation.

rA-G can solve HIFF using a single layer network

Require image of network

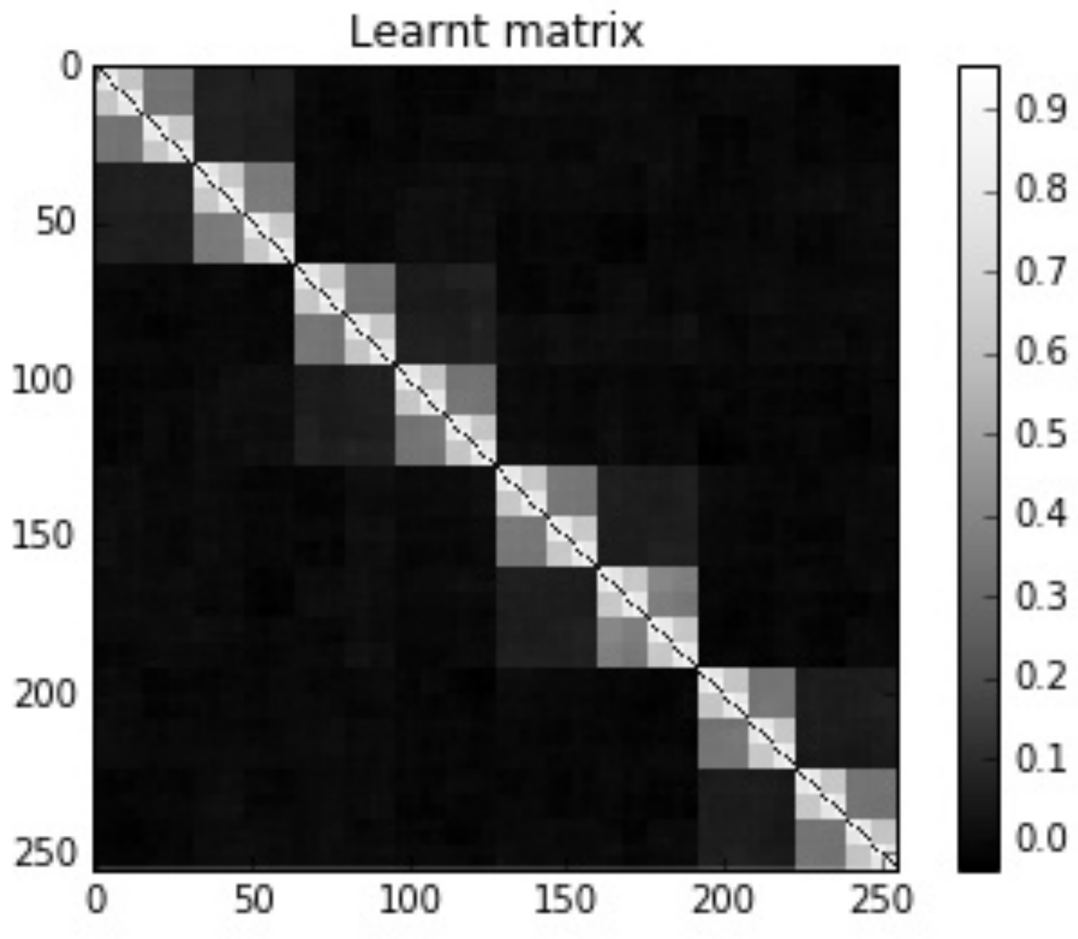


FIGURE 3.13: rHN-G can represent the hierarhical problem structure by differences in the strength of connection between variables.

its not important to maintain the information for how a higher-level representation was constructed in the tree structure. A higher-order unit encapsulates the low-level units encapsulates the variables completely. Thus an organised variation of the lower-units, provided by the high-level unit is sufficient to efficiently explore the solution space. This indicates higher-level units sharing low-level units would require a multi-level representation as a complete encapsulation would cause other higher-level units that also require information about the same lower-level units would fail.

An Autoencoder can learn 1hot encoding, rHN-G cannot it should be theoretically possible to create a function that, when represented by a neural network, requires different weights on different layers – such as representing parity in a feed-forward network.

Inspired by Hinton 4-2-4 Encoder Problem

rHN-G fails to learn the problem structure

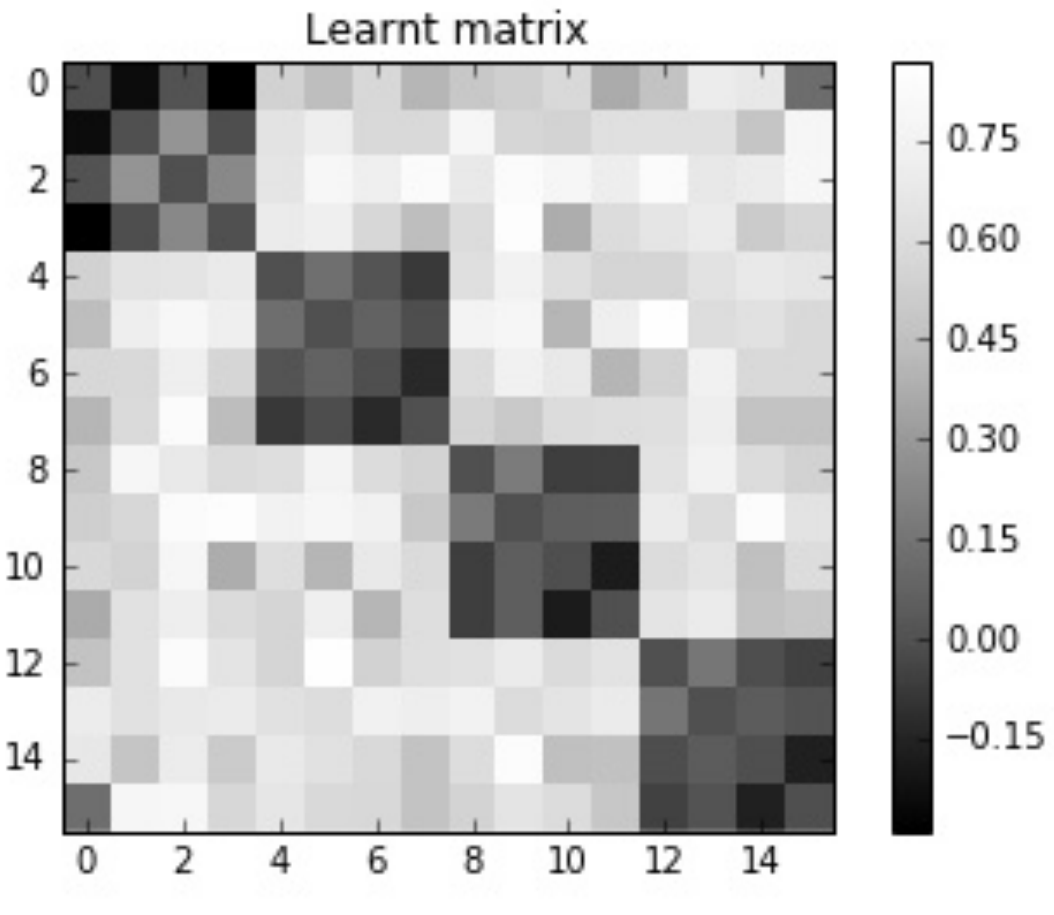
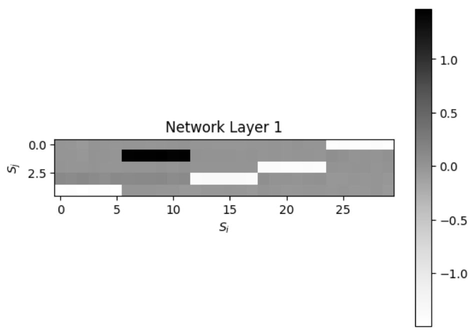
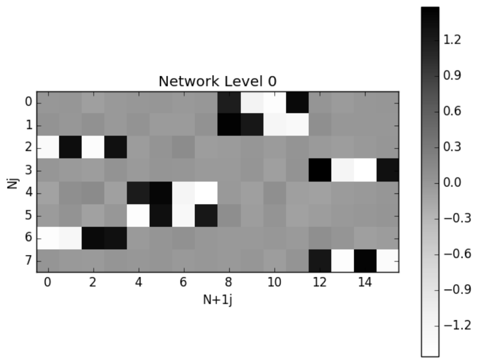


FIGURE 3.14: Given a locally optimal solution, furthest from the global optimum, rA-G is capable of escaping the local to find the global optimum solution.

Figure Figure 3.15 The autoencoder model is capable of learning the 424 encoding for each building-block and correctly separating the building-blocks.



(a) Autoenocder model learning MC problem structure



(b) Autoenocder model learning 424 Encoder problem structure

FIGURE 3.15: Weights learnt by the Autoencoder Model

The comparison between rHN-G and rA-G is demosntrated by a simple experiment where

This provides preliminary evidence that the autoenocder model can corretly represent problem structure that the correlation matrix used in rHN-G cannot.

3.3 Summary

- An autoencoder can replace the correlation model in rHN-G. The method for extracting the information involves encoding the solution, making a local change at the latent representation, and then decoding the change back to the solution level. - Hiff is not a deep problem, even though it has an explicit multi-level problem structure - There does exist problem structure that a NN can represent that a hop field network cannot. - Provide sufficient direction to develop Deep Optimisation.

Chapter 4

The Deep Optimisation Algorithm

In this chapter, the Deep Optimisation algorithm is detailed along with variations to functions that are explored in Chapter 6. First, a short review is conducted that connects the functional operation of Deep Optimisation with the deep learning literature. Namely, inducing a higher-order representation of the data set that improves the computational task. Then, details of the algorithm and implementation for DO are provided. Finally, DO the functionality of DO is compared with the SOTA methods identified in Chapter 2. Due to the algorithmic differences between the SOTA methods, and DO, understanding the performance differences between the algorithms can sometime appear black-box and sometimes algorithms can be excluded from comparison due to differences in the details. However, we argue that functionally the algorithms are directly comparable with each other and DO. This is presented in the final section of this Chapter. The comparison develops our understanding between the functional differences between the algorithms, allows for constructing hypothesis for the types of problem challenges that will distinguish the performance and clarify the contribution that DO makes to the evolutionary computation community.

4.1 Introduction

Section provides the main Concept of DO - high level functional idea, allowing for the reader to then use this to connect with the machine learning literature and theory

DO emulates the process of Evolutionary Transitions in Individuality Smith and Szathmáry (1997); West et al. (2015). Induced representations of a solution correspond to different levels of evolutionary units that are constructed from individual evolutionary units in the layer below. The higher-order evolutionary units provide a mechanism for natural selection to act, facilitating a mechanism for searching at different levels of organisation. The idea of Deep Optimisation relates to the biological organisation observed in nature, and the understanding of evolutionary dynamics provide inspiration for how a multi-level representation is constructed and how the information is exploited. This thesis provides the technical design to emulate the evolutionary process of ETI and apply this to evolutionary computing (search for a solution to an optimisation problem) However, we conclude that this implementation of deep optimisation is not the only way. As we discuss, other models and other methods for exploiting the information can be used. It remains an open question what the correct answer is to the design decisions are. Never-the-less this thesis provides exploration of some of these decisions and guides the direction of future development and research.

Deep Optimisation is the idea of transforming the neighbourhood of a solution, bending and folding the space to remove fitness valleys, allowing for search at this representation to navigate through a reorganised space that has removed the ruggedness of the fitness landscape. A truly unique functionality of DO is the ability to represent multiple layers of representation of a solution, with deep layers providing a higher-level abstraction of the features from the layers below, allow for a truly recursive transformation of the solution neighbourhood.

The representation of a solution can significantly effect the complexity of the fitness landscape Rothlauf (2006), inducing random rugged features into the landscape, and therefore the effectiveness of a local search algorithm. Therefore, by aligning the representation and search operator with the natural problem structure, we can simplify the fitness landscape, allowing for local search to efficiently navigate through the solution landscape.

DO share similarities with the well-known Variable Neighbourhood Search method Hansen et al. (2010), where a fixed set of neighbourhoods are manually defined prior to searching in them. However, in MBOAs, and specifically DO, a compression of a neighbourhood is induced from solutions found in a larger neighbourhood, providing a new and intelligent space to search in.

The adaption to a solutions neighbourhood is provided by transforming the solutions representation. A solutions neighbourhood is defined by its representation and the search operator, e.g. a single-bit variation search operator and binary representation defines a neighbourhood of solutions that differ by only a single-bit from the solution. Therefore, in order to adapt a solutions neighbourhood, the representation, and, or the search operator can be transformed. In Deep Optimisation, the primary mechanism used to adapt a solutions neighborhood is by transforming its representation. This allows for local search (variation and selection) to be applied in a new, and intelligently organised, neighbourhood. However, we also explore how the model can be used to determine the search operator to perform at the higher-level transformation - see section 4.3.2.1

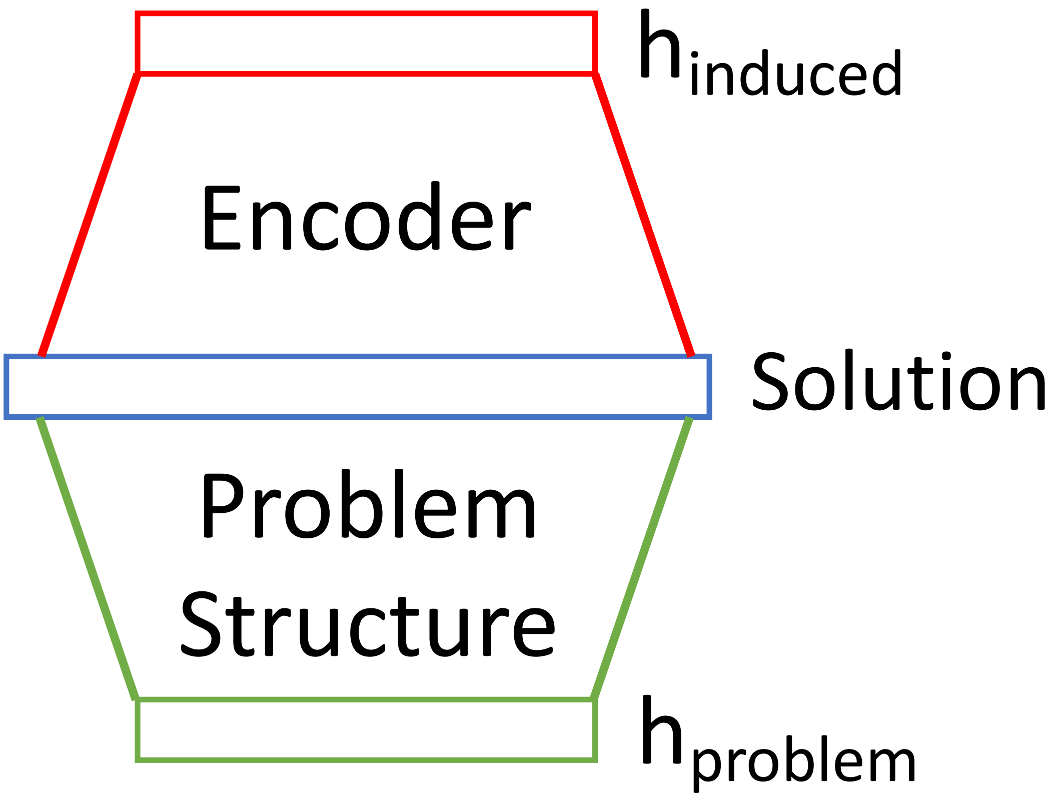


FIGURE 4.1: Inducing a representation that captures the natural structure of the problem via the interface of the solution.

As detailed in Chapter 2, there exist multiple algorithms that have used a neural network in the domain of evolutionary computing, and specifically inline with the building-block hypothesis. However, they are constrained to the algorithm of EDAs. Specifically, using the model to generate a new sample. Their exists no MSSA that uses a deep neural network. Further, in the majority of implementations, only a single layer is used. However, Deep Optimisation if founded on the principle of variation selection occurring at multiple-levels of representation - allowing for search at multiple scales of organisation within a single model. Thus extracting a deep representation of the solution space.

The development of the Deep Optimisation Algorithm can be broken down into three components:

1. Providing a signal of the problem structure to learn from.
2. Inducing a representation that captures the directions of variation in the dataset (variation between partial-solutions)
3. Exploiting the information to guide the search.

Providing a signal to learn (which relationships to learn): In DO, the dataset to train the network (construct the model), is produced by a dynamical process of variation and selection. Providing a signal to learn (which relationships to learn): The model must capture meaningful relationships between variables. The term meaningful relationship is purposefully vague and open to interpretation. In DO, and more generally in evolutionary computation, it is assumed that meaningful relationships are those that regularly contribute to a solutions quality (in many contexts).

Therefore, DO assumes that small, low-level relationships that initially contribute to a solutions quality are used to construct higher-level features and contribute to the global solution. If the globally optimal solution does not contain these relationships, then constructing higher-order components, and ultimately the global solution, from these lower-level relationships will cause DO to fail to find the global solution.

Induction of the relationships (how to represent the relationships): The inductive bias of the model is responsible for inducing a representation that captures the relationships between variables. A good representation is DO is one that allows for local search at this representation correspond to searching in the space of building-blocks i.e., conserving the low-level relationships and searching in combinations of them. At any given representation level, the relative fitness landscape can contain rugged features, causing sub-optimal solutions to be found. In DO a new layer is added to induce a higher-order representation from the deepest representation of the model. Therefore, a unique ability of DO is that it performs a a truly recursive rescailing of the variation. Note - different models represent these relationships in different ways and thus determines the methods available for exploiting this information.

Exploiting the learnt relationships (how to improve the quality of a solution): Given an induced higher-order representation, exploiting this information requires a method for evaluating how movements in this space correspond to movements in the solution quality. In DO, this is achieved by transform movements in the higher-order representation back to the original solution representation, allowing for the fitness differential to be calculated as the solution representation level.

Note that these design decisions are not mutually exclusive. The signal provided to the model should be aligned with the type of variants we wish to capture. The induced representation should be aligned with the method of model exploitation. Finally, the exploitation method should be aligned with the representation of the information and the dynamical process desired with regards to variation and selection.

For most efficient search in the space of building-blocks, it is idealised that the adapted neighbourhood is organised in a way that local variants correspond to individual changes to building-blocks. The solutions to all other building-blocks remains unchanged. Therefore the induced representations presents a search space of partial solutions that is organised such that local variants at this representation level correspond to variation of a single building-block at the original representation. Inducing such a representation is connected with the representation learning research field within the broader field of deep learning. In the next section we review the related literature to inducing a representation that enables higher-order search.

4.2 Literature Survey

In this section we review the relevant literature regarding deep learning and how this connects with the Deep Optimisation Algorithm.

Representation learning – how the representation relates to what we want.

Neural network models provide several advantages over other machine learning models, they are flexible, multi-variate models that have efficient learning algorithms that can be run in parallel

To improve the search of a solution requires accessible and meaningful information captured by the model.

A neural network approximates a function by using a collection of artificial neurons (or nodes) that compute a weighted sum of their inputs. A single neuron of a neural network can express a decision surface. In combination, multiple decision surfaces work together to compute the desired output of the function. Generally a neural network is organised into an input, hidden and output layer. A layer referes to the hidden nodes and the weights of the inputs into the nodes, except for the input layer that only contains the input nodes. In order to separate into layers the hidden nodes must use a non-linear function on the weighted sum. A deep neural network is a model that uses multiple hidden layers.

The nonlinear units allow for representing a curved surface to capture the dimensional reduction of the original space. Where as pairwise correlation models, such as that in rHN-G that does not use non-linear activation units, are restricted to planes.

Machine learning encompasses a class of algorithms that learn an approximation to some function. In neural network models, this function is controlled by the objective function that . - Neural network models have been used for a variety of tasks. - generation, classification, regression and dimensional reduction - Shows the usefulness for extracting different types of features, from different types of data to solve different types of tasks. In classification the perception model the line that separates the variables into there respective classes - representing the decision surface that separates the classes. In dimensional reduction the perception transforms the original space, by stretching and squeezing the space such that salient features are represented by fewer units. For example, in classification, learning that 00 and 11 are useful combinations also requires learning that 01 and 10 are not useful. Thus a decision boundary separates the two sub-sets. In dimensional reduction, the combinations 00 and 11 will be represented by transformed into a lower-dimensional space and represented by a single variable, thus 01 and 10 may not have meaning in this space. these combinations closer together by reducing the space fro 01 and 10.

- Of significant interest with regards to this thesis is to understand how a deep neural network affect the performance, with specific regard to multi-scale search. The concepts of searching at multiple scales or organisation is not well understood, however, intuitive a deep neural network natural fits the idea. Frequently used examples of what a deep neural network can provide is - multi-level feature extraction - extracting higher level features from the data set (often observed in imaging) Zeiler and Fergus (2014) - non-linear relationships between variables, XOR example Minsky and Papert (1969). A historical example of the limitation of a single layer network was presented by Minsky and Papert (1969). They demonstrated that the single-layer perceptron cannot solve problems which are not linearly separable – the classic example being the exclusive-OR function (XOR). In order to do so, a multi-layer feedforward network is required. The hidden layer represents a space a latent variable that underlies the mapping between input and response variables. As such the hidden layer of computation is a mapping performs a nonlinear separation of variable's and represents the inputs using a latent variable. The space of latent variables provides an approximate linearly separable representation of the nonlinear re-presentable input nodes. The next layer, between the latent variables and output variable(s) can correctly separate the inputs to match the desired response – the hidden layer computes the relationship between the latent variables (representing correlations of input variables) and the response. It is proven that neural network with a single hidden layer using logistic activation function (only necessary at the hidden variables) can approximate a bounded continuous functions Hornik (1991)

By introducing and developing a successful algorithm that uses a DNN models opens up many further directions for research. The tool-set for deep learning is vast, from different activation units, designing objective functions, regularisation's such as drop out, memory and attention mechanism, optimisation learning methods, different architectures such as Convolutional Neural Networks (CNN) for transnational invariant features and recurrent neural networks suitable for sequence data. In addition, there is a wide range of research focusing on ideas for transfer learning, one or few-shot learning. Thus by making the connection between Deep Learning and Evolutionary Algorithms, provides a research direction to exploit all these advanced ideas for the use in black-box optimisation.

Connecting deep optimisation and the theory of representation learning - What we need the model to do - - Perform a dimensional compression of the solution space by learning relationships between variables using only the solutions (unsupervised learning). - - Separate and represent partial-solutions that allows for individual combination of building-blocks - - Extract information for partial-solutions from the model

Partial solution learning - the model must be capable of capturing the relationships between partial-solutions and adequately separating them from alternative partial-solutions to allow partial-solution substitution for a solution

Hierarchy - The idea of hierarchical organisation in systems and multi-scale search fit naturally with the idea of deep learning, and specifically hierarchical feature extraction from a data set - The idea of different scales of evolutionary (search) units shares similarities with the idea of deeper (higher-level) hidden units in a neural network representing higher-orders of organisation (evolutionary units) - This ideal organisation is not guaranteed in deep learning.

The idea of extracting and separating features and also constructing a hierarchical organisation of features is a focus of representation learning in deep learning.

-The idea of representation learning provides us with away of think about the performance of a deep learning model. In the next section we review the theory behind inducing representation to provide an understanding for how to correctly implement a neural network model and which neural network model will be suitable for deep optimisation application.

4.2.1 Representation Learning

In this section, we discuss the field of representation learning in deep learning and connect the ideas with exploiting building-blocks in evolutionary computation. This review aims to provide connections with:

1. How a neural network learns the problem structure and represents this. i.e., where is the information stored
2. How the information can be exploited.

What makes a good representation.

1. Provide the same information as the original data
2. Robust to noise - small perturbations away from the contributing relationships
3. Represent and separate salient features - disentangled representation of factors of variation in the data-set.
4. Allow for simplifying the computation necessary to meet the objective.

It is not guaranteed that inducing a such a representation will be relevant to application. However, for exploiting the bu idling-block hypothesis and near-decomposition, this provides a suitable domain to explore explanatory factors/features of the data distribution

What is representation learning Representation learning corresponds to extracting meaningful features from the data-set, and representing these features in a meaningful way. Of course, the vagueness in the use of meaningful and features allows for an open interpretation for what exactly representation learning is. However, it is best described as constructing or extracting a representation of the data-set that can be interpreted in a way that makes computing the function, such as classification, more straight-forward.

- Modelling, or performing computation in the original space can be difficult due to the high dimensional, there for transforming the representation to a simpler space allows for more efficient computation for the desired task.

In Deep Optimization, the task of the model is to extract partial-solutions from the distribution of promising candidate solutions. This requires correctly identifying and capturing the relationship between variables that form a building-block, but also seperating the building-blocks to allow for individual recombination via some method to exploit the information.

Dimensional reduction provides an unsupervised method for extracting salient features from the data-set. The induced feature set is dependent on the inductive bias of the model used to identify, represent and separate building-blocks.

- linear transformations, translation, scale, shear. - Nonlinear transformations – also twist the space

4.2.1.1 Feature Extraction - Manifold

Feature extraction - representing features that describe the input. Invariant features are robust noise and thus insensitive to small variances from the feature.

What is a manifold Manifold assumption - high dimensional data is only artificially high, the data actually lies on a lower-dimensional manifold Cayton (2005). ‘A data point can be described as a function of a few features’. ‘Manifold learning algorithms attempt to uncover parameters in order to find a low-dimensional representation of the data’.

Concentration of relationships between variables concentrate in regions of lower dimensionality that the original space

How does this relate to learning building-blocks

building-blocks refer to variable combinations that contribute to a solutions quality. Therefore relationships between variables occur regularly and therefore, in a distribution of solutions, relationships concentrate to a region of smaller dimensionality.

How can we learn manifolds Dimension reduction a process of learning the underlying or hidden relationships between variables that define the dataset.

Learning the factors of variation.

Nonliner representation can unfold of nonlinear manifolds to a linear representation - a linear movement at the representation corresponds to nonlinear movement in the folded manifold.

Feature engineering – in a way features are like basis function that are supplied naturally by the data.

Attribute-specific data manifolds: there might be different manifolds for each attribute

Assume that the data lies on an embedded non-linear manifold within the higher dimensional space.

4.2.1.2 Disentanglement

What is disentanglement Brahma et al. (2015) - exploring why deep learning works from a manifold disentanglement perspective and learning a good representation of the data makes it easier to use the information for the task, such as classification, generation. In the case of DO, for search.

disentangling as many features as possible from the dataset such that as little information of the input is lost at the new representation (feature representation)

- Most important aspect is not what determining what the model captures and ignores (invariant features) but rather, how we can seperate different features from each other.

-Disentangled representation learning is an unsupervised learning technique that separates, or disentangles, each feature into narrowly defined variables and encodes them into separate dimensions. In disentangled representation, a single node can learn a complete feature independent of other nodes. For instance the rotation of an object.

-Disentangling the features of variation observed in the dataset. - cause by an entanglement of many manifolds in a data set

-Disentanglement aims at extracting the salient features and representing a disentangle representation of entangled of manifolds in the original space - separation of the salient features that vary between inputs (factors of variation). Therefore variation between inputs can be described as independent changes to these features.

How the idea of disentanglement relates to building-block Learning a disentangled representation directly refers to extracting and separating features from the input such that variations in the input are represented as changes to individual features.

For exploiting the BBH this is the desired functionality. A representation that allows individual and separate variation of features (building-blocks). Therefore inducing such a representation leads natural to search in the space of building-blocks by applying variation to the extracted features.

Learning a representation of features that are separable that allows for individual recombination

Learning a disentangled representation suggests that the usefulness of a neural network is in its ability to separate entangled factors of variation. Extrapolating to building-blocks indicates that a neural network model will be sufficient for learning a disentangled representation of entangled of building-blocks, or learning a representation that allows separable variation of overlapping building-blocks.

How can we learn a disentangle representation

4.2.1.3 Distributed Representation

What is a distributed representation Localist representation - a single neuron represents one thing, for instance a building-block - inefficient when data has complicated structures. Distributed representation - many to many relationships between representations. A feature in a dataset is represented by many neurons.

How does this relate to building-blocks A building-block can be represented by multiple, hidden neurons. Multiple neurons to represent a building-block not been explored - for example in HIFF, a higher-order feature is represented by a single neuron

What have we learnt from the theory of representation learning and how does this help use with developing Deep Optimisation - Thus the theory in machine learning, and in particular the ideas that is used to understand how neural network models work and what they can do provides an inspiration for the types of structures or relationships in the problem structure that a neural network model will be capable of dealing with that other, less sophisticated models, will not be able to deal with.

4.2.1.4 Deep Representations

Deep representations allow for

1. reusing low-level features to construct higher-order features (overlapping dependencies between building-blocks)
2. related to the idea of efficient representation of higher order features.
3. extracting more abstract features at higher-level representations

Learning a deep representation remained a challenge until a breakthrough paper by Hinton and Salakhutdinov introduced greedy-layer-wise pre-training to construct a deep representation Hinton and Salakhutdinov (2006).

A deep representation of the data set is induced by using an unsupervised learning to extract features from the data set one layer at a time. A deep layer would extract features from a lower-level representation. A deep model is iterative constructed by adding a deep layer once the immediate lower level representation has been learnt. The induced representation was then fine-tune to meet the objective function, producing SOTA results.

This shares connections with out interpretation for emulating the transitions in individuality: - The lower layer providing clarity on the signal to learn at the next layer - Without lower-level components, its not feasible to construct higher-order components - evolution does not have the foresight, or abstraction, the provide a direction of where it wants to go.

Summary - what has this brief survey done for us An open question in deep learning research is how inducing a good representation helps with the task of learning Bengio et al. (2013). For optimisation, the same question arises, does inducing a higher-order representation that extracts the features (building-blocks) from a solution help further search. It is hypothesised that the answer depends on the problem structure and how well aligned the inductive bias of exploiting relationships that contribute early to a solutions quality also contribute to the global problem.

4.2.2 Auto-encoder Model

Many architectures - we decided to use the autoencoder model due to unsupervised, success, deep construction and representation and versatility - same architecture used in different ways (interpretations) - allow for flexibility for MIG and MIV

- In the field of deep learning there are many models that fit the category. - To satisfy the design decisions require for DO, the Auto-encoder model is most appropriate - - Unsupervised training - - Deep network construction - - Flexible and Generative interpretation - - Simple architecture - - Widely used

- As we have discussed, representation learning presents a foundation for thinking about how a neural network model can be used in the context of learning building-blocks from a distribution of solutions - The autoencoder model is a simple example of representation learning - The autoencoder model has also been used to construct multi-level representations, via the concept of stacking - Hinton and Salakhutdinov (2006).

The autoencoder architecture has been used as a deterministic model for dimensional compression and also as a probabilistic model for generating new samples Kingma and

Welling (2013). Therefore the autoencoder model allows for both a probabilistic and deterministic interpretation, making it suitable for MIG, MIV, and inducing different representations. This allows us to explore different design decisions using the same more architecture.

- - Encoder network leans to extract features from the input, represented by the hidden layer. $h = E(x)$, where h represent the feature vector of input x, computed by the encoder network E. - - Decoder network D learns to reconstruct the input x_r from the feature vector. $x_r = D(h)$

- Encoder does inference: interpret the data at the abstract level (Marginal independence in latent space)

- By learning a network that can reconstruct the input from a high-order representation requires the encoder to extract the most salient features of the input. To enable generalisation between different inputs (or different variants) requires the separation of the salient features (disentangling the features of the data-set).

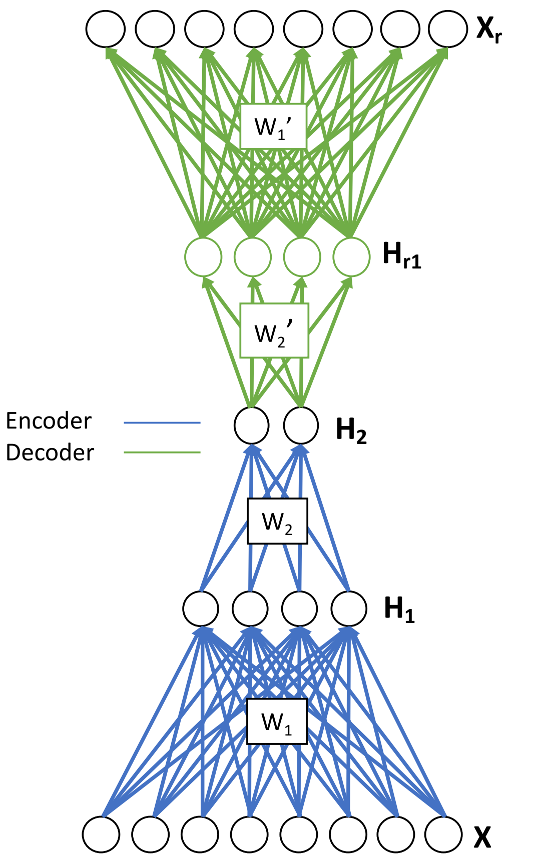
- By constructing a deep model provides the capacity to represent higher-order abstractions of the data set, where higher-order features that describe the data set are constructed from lower-order features. Hinton and Salakhutdinov (2006) introduce a efficient method for training a model to extract a deep representation.

Auto-encoder Architecture An auto-encoder is a relatively simple feed-forward neural network architecture that is trained to learn to output a reconstruction of the input.

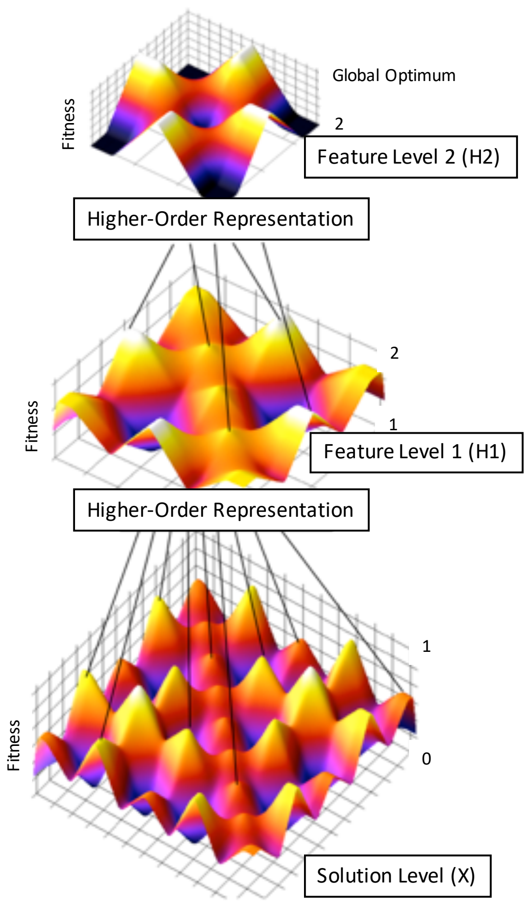
The autoencoder model is within the class of unsupervised learning models.

Figure Figure 4.8(a) presents the architecture of autoencoder model with 2 layers. X is the input to the network and X_r is the output. The Encoder network transforms the input to a lower-dimensional, compressed representation of the input. The induced representation is the activation response at a hidden layer. A deep autoencoder consist of multiple layers of hidden representations. The example presented in Figure Figure 4.8(a) consists of two layers. The output from the 2^{nd} layer, H_2 is called the bottleneck layer. The code at the bottleneck, calculated from the input using the encoder network, is used as the input to the decoder. The Decoder network transforms the latent representation back to the original representation. Because the latent representation is of a lower-dimensions of the original solutions space, the decoder is performing a lossy reconstruction of the input, relying on only the salient features in input to reconstruct the whole input.

By applying constraints or regularisation's on the network, such as having a lower dimension of units at the hidden layer compared to the number of input units, The autoencoder model is encouraged to learn a compressed representation whilst maintain a good reconstruction of the input. This induces a latent representation that captures the most salient features that contribute to the reconstruction of the dataset.



(a) Architecture of an autoencoder model with 2 layers



(b) Illustration of the transformation made to the fitness landscape caused by the feature extraction

FIGURE 4.2: Example of a two layer autoencoder model

Examples of layerwise training Semantic Hashing - placing semantically similar data next to each other in the training set. Salakhutdinov and Hinton (2009). Thus local changes in this representation would translate to a local search in the high-dimensional of the original space and not random recombination.

DO uses an autoencoder model to learn a higher-level representation of a solution. The autoencoder model consists of an encoder (E) and decoder (D) network that transforms solution (S) to a latent representation (H) and then back to the original input representation. Specifically, $S_r = u(X_r) = u(D(E(S)))$, where S and S_r are the solution and solution reconstruction respectively, X_r is the output from the decoder and u is a unit step function. Specifically,

$$u(X) = \begin{cases} 0, \text{ if } X < t \\ 1, \text{ if } X \geq t \end{cases}, \tag{4.1}$$

The encoder network performs a transformation from the visible units, X, to hidden units H_l. The model is composed of L hidden layers, each transforming the lower representation H^e_{l-1} into a high-order representation H^e_l, where H^e_l represents the activation response of layer l in the encoder network (e). This transformation is achieved by using the hyperbolic-tangent nonlinear activation function (f) on the sum of the weighted inputs. Specifically, $H^e_l = f(W_l D_r(H^e_{l-1}) + b^e_l)$ where W_l and b^e_l are the weights and bias respectively for encoder layer l. D_r is the dropout function. At the first hidden layer ($l = 1$), $H_{l-1} = S$ (the encoder input).

The decoder network generates a reconstruction, X_r, from the latent representation at H^e_L, often referred to as the bottleneck layer for the autoencoder. The decoder network contains the same number of layers as the encoder network. Each layer of the decoder transforms the hidden representation H^d_l back to a lower-order representation H^d_{l-1} via $H^d_{l-1} = f(W'_l H^d_l + b^d_l)$, where W'_l is the transpose of the encoder connection weights W_l (the weights are tied), b^d_l is the decoder bias and H^d_l is the decoder (d) activation response at layer l. At $l = L$, $H^d_L = H^e_L$ (the input to the decoder network is the output of the encoder network) and at $l = 1$, $H^d_1 = X_r$ (the output of the decoder network is a reconstruction of the input S).

Effect of tied weights - (fewer parameters, regularisation on the weights) - Encourages the hidden neurons to saturate - to maximise the reconstruction - encourages the hidden units to represent perpendicular directions of variation in the dataset (need citation)

The autoencoder model produces an output, which is the same as the input, by decoding (reconstructing) the encoded (dimensionality reduced) information. Without an internal transformation process such as dimensionality reduction and expansion, the model can learn converge to an identity matrix that can easily achieve the objective function.

The weights are initialised using the uniform distribution $U(-0.01, 0.01)$. Both the encoder and decoder biases are initialised to 0. In this paper, DO used the TanH activation function. Therefore, at training, a binary solution $[0, 1]^N$ of size N is transformed into the discrete representation $[-1, 1]^N$. The decoder output is in continues space, bounded by [-1,1], and converted to a binary representation using the unit step function $S_r = u(X_r)$.

The size of each layer can be manually assigned. The compression factor, r, is the ratio between the number of input to output units at a given layer. Specifically, $N_l = rN_{l-1}$, where N_l and N_{l-1} are the number of units at the input and output of the layer l. A compression ratio.

4.2.2.1 Regularisation

Regularisation and objective functions are designed to encourage particular characteristics in the induced representation, and encouraging the model to learn more interesting structure that just the identity function.

The type and degree of regularisation applied will effect the inductive bias of the model.

Compression Ratio - Under-complete (by limiting the capacity) - Compression ratio is the change in the dimensionality of the representation between a layer. Too strong a compression ratio reduces the capacity of the model and therefore is unable to capture the relationships between the variables. To weak a compression, and or expansion, increases the risk of learning the identity function. *L1 Regularisation* - Encourages a representation that minimised the total sum of the absolute weights. Causes a bias towards a sparse representation where neurons respond to few inputs. Connections that do not provide a significant contribution to the objective function (significance controlled by the lambda term) are penalised. *L2 Regularisation* - Encourages a representation that reduces the maximum strength of connections, making the weights smaller in absolute value. This reduces the risk of overfitting reducing the liklihood of the network weights becoming over specialised to capture small changes in the dataset.

Sparsity Regularisation - Encourage a neuron activity to respond infrequently - cite

Contractive Regularisation - Encourage the Jacobian of the weights - Rifai et al. (2011) Sampling from a CAE Rifai et al. (2012) -

4.2.2.2 Denoising

Denoising autoencoders refer to models that learn to reconstruct the original input from a corrupted input. The corruption is applied directly to the input and can take various forms Vincent et al. (2008, 2010). Learning a model that captures the relationships between variables, such that during reconstruction, missing information, or corrupted information can be filled from the value of other variables (or the feature) and the the relatinoships teh currupt variable has with that variation (feature).

Dropout Hinton et al. (2012b) is a more general application of noise. Dropout removes any signal from the units applied to. Dropout can be applied to any neuron in the network. It promotes the model to learn how to reconstruct the input from missing date - encouraging the model to capture relationships between variables such that the value of the units dropped out can be reconstructed from the state of other variables and the relationships between. This provides a signal to the learning algorithm of the underlying relationships that construct the dataset.

Dropconnect Lee et al. (2007), similar to dropout, encourages the model to learn a robust representation. It encourages the model to learn a robust representation where connections between visible and hidden units are unreliable.

When using a denoising autoencoder the compression ratio of the model becomes, to some degree, redundant when the model is tasked to reconstruct the input from unreliable data (noisy inputs).

4.2.2.3 Variational

Disucss why not using variational autoenocder - not sure neccesary

The Variational Auto-Encoder (VAE), Kingma and Welling (2013) which is a well-known probabilistic interpretation of the autoencoder model

- The VAE is a generative model - the latent space can be sampled to generate new data points. The autoencoder model does not provide such a model. - For MSSA, it is not a requirement that a solution can be generated from the latent space. Instead, we require that features can be extracted and exploited. - VAE use a prior distribution (such as Gaussian) of the latent response. The performance of the reconstruction for the data with a high variance is lower than that of a general auto-encoder. Therefore, we adopted a general auto-encoder. - Inducing a disentangled representation is not performed well by VAE's Kumar et al. (2017)

- VAEs have a strong regularisation pressure to create a representation that is amenable to sampling. That is we can generate a new data point by sampling the latent space using a Gaussian distribution. However, this reduces the representational power of mode and does not lend its self naturally to a recursive application and learning a multi-level representation that is use-full for independent variation. Never-the-less it remains an ongoing area of research Zhao et al. (2017)

VAE still challenging Ghosh et al. (2019) Stacking VAE not well developed, although progress is being made Zhao et al. (2017)

4.2.2.4 Deep Autoenocder

End-to-end Layer-wise Stacking Vincent et al. (2010) – plus fine tuning Hinton and Salakhutdinov (2006) Layer-wise Building Jain and Seung (2008)

4.2.3 Summary

- We have introduced the theory of representation learning in neural network models and how this corresponds to learning building-blocks in optimisation problem - We have discussed how the autoencoder model provides a suitable architecture to emulate ETI and be used in a MIV and MIG way. - Further, the using the autoencoder model provides a suitable foundation in order to devlop DO further.

4.3 The Deep Optimisation Algorithm

Need to discuss how one layer provides a signal to the other The interaction of selection on one layer, providing a singal of features for the next layer, whilst providing a negative feedback for spurious feature Better Description of the algorithm - how it was built-up to the final algorithm For each main functional decision, include evidence that the algorithm is doing that

1. MIV - walk through the encoding decoding process
2. MIV - demonstrate a trajectory
3. Induction - demonstrate that information is being represented at the hidden layer.
4. Induction - Encoding provides a new representation
5. Multi-level search - schematic of how one can search at multiple levels of representation

In this section we detail the deep optimisation algorithm

The Deep Optimisation algorithm is presented in Algorithm 5. The algorithm consists of two optimisation cycles, a solution optimisation cycle and model optimisation cycle, inter-locked in a two-way relationship. The solution optimisation cycle is an iterative procedure that produces a locally optimal solution using model-informed variation. The model optimisation cycle is an iterative procedure that updates the connection weights of a neural network to satisfy the learning objective.

Algorithm 4: The Deep Optimisation Algorithm

```
Initialise Model while While Termination Criteria not met do
    while ModelPerformance == Poor do
        for Solution in Solutions do
            Perform Model-Informed Generation;
            Perform Model-Informed Variation;
            Perform Replacement Strategy;
        Training Set = Solutions;
        Update Model with Training Set;
        Measure Model Performance ;
    Perform Model Transition;
```

DO uses the deep learning Autoencoder model (AE) due to its ability to learn higher-order features from unlabelled data. Further, the autoencoder model allows for layerwise construction, building a deep representation, one layer at a time.

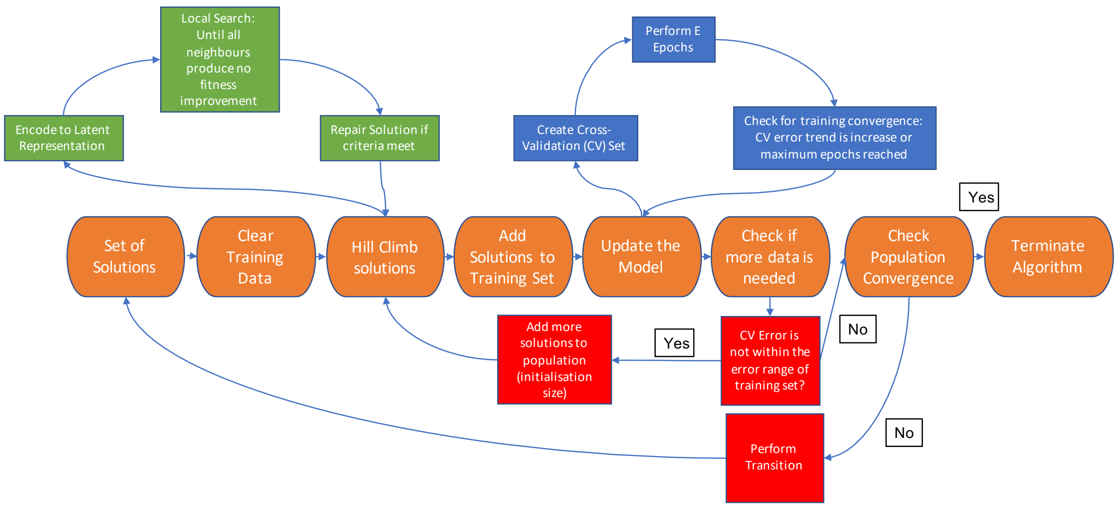

Local Search: Until all neighbours produce no fitness improvement
Encode to Latent Representation
Repair Solution if criteria meet
Perform E Epochs
Create Cross-Validation (CV) Set
Check for training convergence: CV error trend is increase or maximum epochs reached
Yes
Set of Solutions
Clear Training Data
Hill Climb solutions
Add Solutions to Training Set
Update the Model
Check if more data is needed
Check Population Convergence
Terminate Algorithm
Add more solutions to population (initialisation size)
Yes
CV Error is not within the error range of training set?
No
No
Perform Transition

FIGURE 4.3: Schematic of the DO algorithm

4.3.1 Solution Representation

I want to use this section to first illustrate how simple the idea of DO is - search is just performed in a new space of partial solution - so explain what a partial solution is - How we identify a partial solution, and therefore what the induced representation, represents.

Then Technical implementation details

Then Illustrate how the neighbourhood is adapted - - Figures: the avaliable solutions - Interpolation between solutions

I want to be clear, that it is hierarchical consistent. What applies at the solution level, applies at any hidden representation

- This section describes how a solution representation is transformed - It is hierarchical consistent, local search is performed at different levels of representation - This representation is induced by the model - The representation is encouraged to represent relationships between variables that contribute to a solutions quality - provided by the selection pressure and the inductive bias of the model

What is a partial solution - a partial solution is a particular combination of variables, or a relationships between the variables, that contribute to the quality of a solution. Small, low-order partial solutions are confined to only a small number of variables, whilst high-order partial solutions can refer to the entire solution.

Identifying these partial solutions is a key challenge for DO and other MBOAs. A partial solution can be identified as a regular occurring relationship between variables in a distribution of promising solutions (i.e., some selection pressure has been applied to the distribution of solutions). Therefore, these relationships are regular in many different solution contexts.

Need to include more about the idea - learning from previous experience to guide future variation. i.e., regular variable combinations that contribute to a solutions quality are assumed to be useful for the complete solution. Therefore, rather than repeatedly finding these combinations (partial solutions) we conserve them during future search, assuming that these combinations will continue to be useful in other solutions contexts.

At initialisation a population of solutions is generated from a uniform distribution $U[0, 1]$. Local search is applied to each solution using a single-variable variation to produce a population that is locally optimal relative to the solution representation.

Figures to demonstrate the DO induces a representation of partial solutions.

Figure Figure **??** - Available solutions at the latent space

Figure Figure **??** - Interpolation differences between the latent space and the original space

Figure Figure **??** - Difference in the neighbourhood. Local neighbours in the original solution, local neighbours at the new representation. - difference in the hamming distance

DO utilises a layer-wise approach for both training and generating samples. Initially the AE has a single hidden layer and is trained on solutions developed using the naive local search operator.

At each hidden level, an optimised model will contain a meaningful compression of the lower level relating to higher-orders of organisation.

The fitness value of a solution is not used for training. The solution will contain variable combinations that contribute to the solutions quality, due to the dynamical process of variation and selection. DO does not attempt to model the fitness landscape in order to predict regions of higher fitness, unlike surrogate model methods. Instead, DO learns a compressed representation of the solutions space, allowing for a more intelligent variation. The model is not learning how to improve the solution. The model is presenting a higher-order solution neighbourhood for variation and selection to navigate through.

4.3.1.1 Deep Model

Unlike MBOAs, which disregard their constructed model after the information has been exploited, DO instead builds on top of the previous model by adding an additional layer to the network.

This not only increases the capacity of the network, but also maintains the lower-level relationships information for how the building-blocks, that have been used during variation and selection to produce the new distribution of solutions. Therefore we hypothesis that this will provide a clearer signal for inducing a good representation of the relations ship and also facilitate search at multiple levels of representation (for low-level relationships to high-level relationships) during optimisation. Something that no other algorithm performs.

Further, it is less efficient to discard previous information about the problem structure and construct a new model.

Further, the low-level relationship's can be valuable information for search later in the process of optimisation. By continuing to represent this information, then this can be exploited at anytime during optimisation.

Learning a good representation is computational costly. A disadvantage of MBOAs is the computational cost in induction compared to algorithms that do not use a model

For DO, it is natural to use the lower-level information capture by the autoencoder. A new, deeper layer can learn to combine lower-level features or induce a higher-order abstraction of the relationships.

Models used by the SOTA do not have the concept of depth in the same way as DO

We discuss the significance of constructing deep representation in Section **??** [] and demonstrate this empirically in Chapter 6

Transition is a term used to define the moment when the bottleneck layer has model the training data sufficiently well that this information can be exploited.

Determining when the model has learnt a good representation of the training data is not straight forward. As we have discussed, the definition of a good representation is vague. We empirical explore what constitutes a good representation in Chapter 6 and methods that can help to encourage a suitable representation.

Figure Figure 4.5

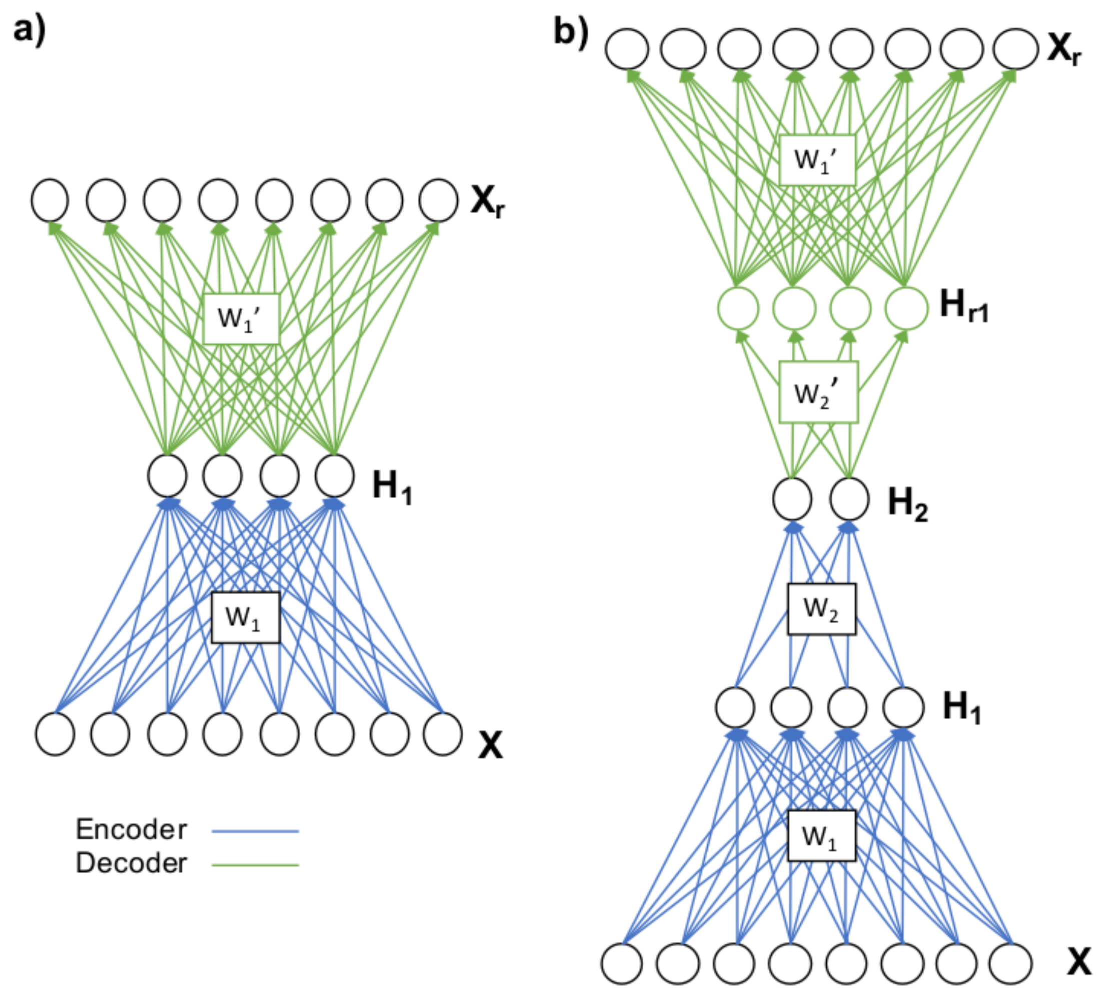


FIGURE 4.4: Networktopology of the autoencoder

4.3.1.2 Training

Training updates the model parameters of the autoencoder model using the distribution of solutions found when hill-climbing in solution representation induced by the current model (local optimum solutions in the neighbourhood at the bottleneck representation).

These solutions are then used as the training set for the autoencoder model with a single hidden layer ($L = 1$). For deeper layers, training is performed using an autoencoder model that has an additional hidden layer compared to the model used for MIV.

Training is performed using the back-propagation algorithm and Adam optimiser to minimise the training cost function, Equation 4.2. The training cost function is an additive combination of the mean squared error (between X_r and input S), L1 and L2 regularisation. λ and γ are the regularisation coefficients used to control strength of the parsimony pressure applied to the model. n is the number of training examples used in

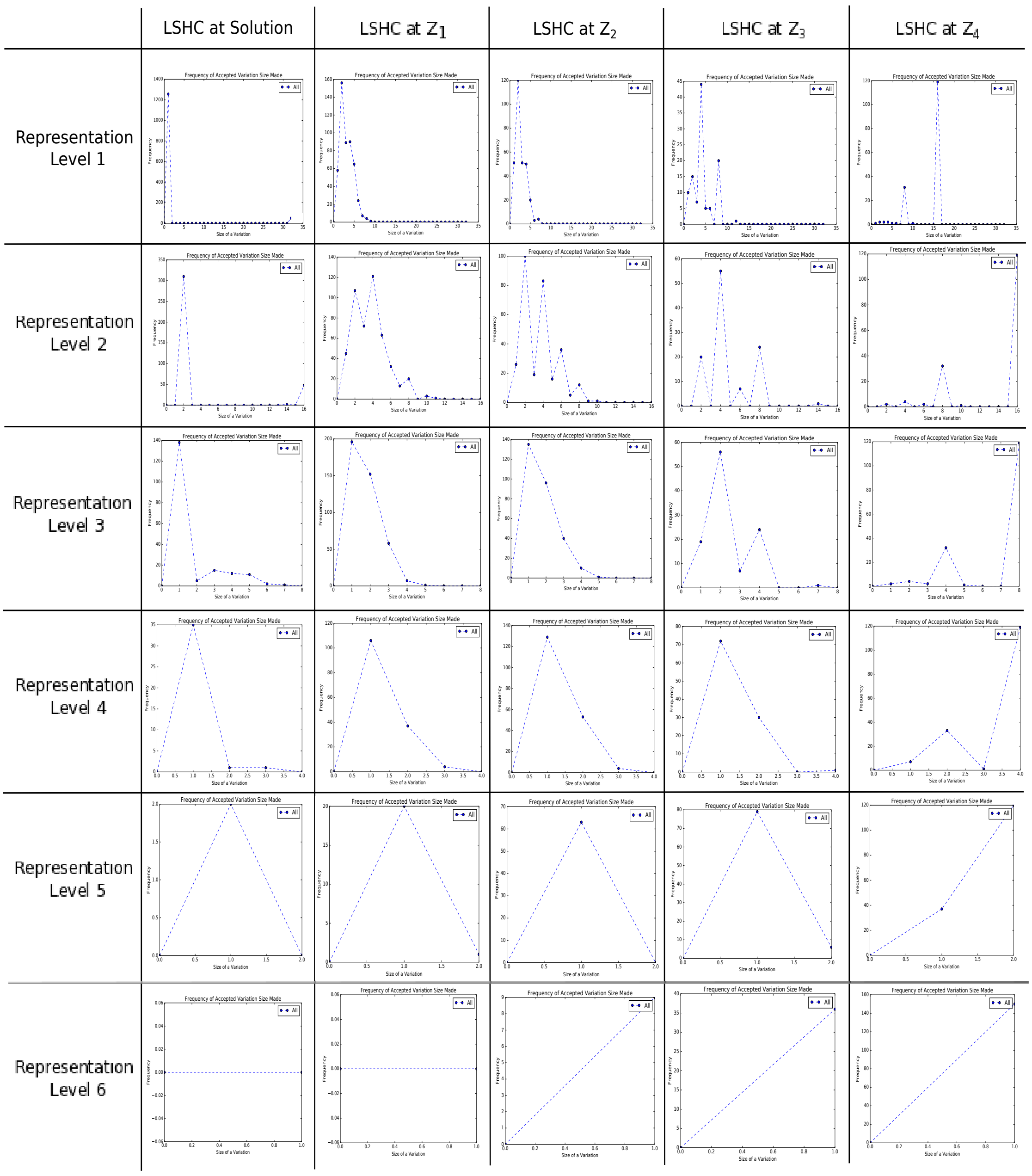


FIGURE 4.5: Illustration the effect of searching at induced higher-order representations (row) of HIFF (size 32) that allow for local search at these induced representation to make large, simultaneous and coordinated variation in the original solution space (row 1).The column represents different levels of the HIFF representation (at representation level 6, there is only a single constraint to satisfy between two building-blocks of size 16. The row represents different layers of the model that local search is performed in.

a mini-batch. All parameters of the model are updated during training.

$$\frac{1}{n}\sum_{i=1}^{n}(S_i - D(E(S_i)))^2 + \lambda\sum_{l=1}^{L}|W_l| + \gamma\sum_{l=1}^{L}||W_l||^2 \tag{4.2}$$

By updating the parameters of the mode to learn compressed representation that facilitates good reconstruction of the dataset causes the model to extract the salient features. Therefore, rare or unseen variable combinations are removed from the induced representation. Therefore the reconstruction error for unobserved data points that are not from the same distribution from the training set. This is demonstrated in Figure Fig:DO$_{R}econError_{S}ameDiffDistribution.Lowreconstructionerroronunseendatapointsfromthesam$

The randomly generate data points show a higher reconstruction error throughout training, whilst solutions from the same distribution (local optimal solutions found by local search), but not used for training, show a reduction in the training cost. We can therefore see that learning is inducing a representation that captures the relationships between variables.

Demonstrate the performance of training

Figure Figure **??**

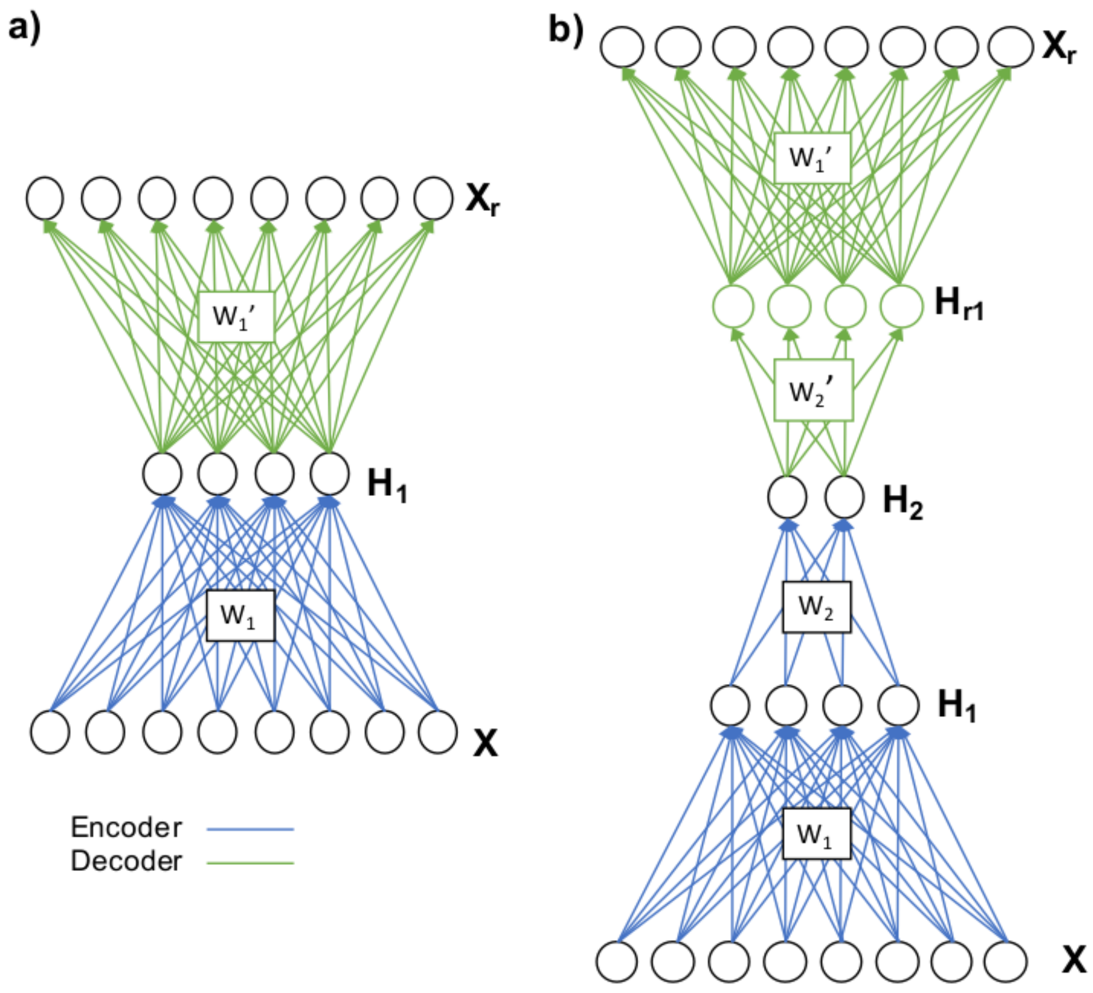


FIGURE 4.6: Reconstruction error for solutions generated at random and solutions found using local-search (and not used for training) during training of the autoencoder model.

Online Incremental learning - rHN-G like

Online learning is the process of updating the model with a single training example and discarding the training example. An advantage of this method is that transitions will happen automatically - a poor/under-defined representation will not provide an update to a solution. Postive, selected for updates, will only occur when the representation extracted meaningful relationships. A limitation of the online method is the learning rate must be small. Making strong associations to early can inhibit the model's ability undo these relationships, therefore rare but important relationships are unlikely to be captured.

A small learning rate is not most efficient for learning when the structure is relatively simple. For instance, in the case where weights require only strengthen to improve the reconstruction of a solution.Because of the small learning rate, this will require many training examples and therefore generating a new solution each time, contribute to more function evaluations. Here a new solution provides no new structural information to the model, costing unnecessary function evaluations. The generation of a solution is wasted. This process can therefore make it less competitive with alternative methods, not because the model could not learn the structure, but due to the gradient based method to update the parameters of the model.

Batch Population based - distribution - EDA like and BBHC like - show results later chapter. To overcome the limitations highlighted in the online learning case, we can reuse solutions. Batch mode allows for the parameter Epochs to be set. In the example above, solutions can be repeatedly used to move the weights in the correct direction without incurring the cost of generating new training examples. The drawback is the tuning the parameter to ensure the training set size contains sufficient information to capture the structural information of the training set and not overfit to a sub-sample. Naïve:

The batch operation of deep optimisation involves first creating a training data set. Random batch of solutions are then used for the back-propagation algorithm. Here, we can take advantage of multiple presentations of the same solution to the training phase. Batch learning comes with the additional hyper parameters batch size and epochs (number of passes through the data set). In the naive implementation, this is specified by the practitioner. An empirical recommendation that has worked well for our experiments is a batch size of 1

Model Hyperparameters The hyperparameters that can be tuned to improve training of a neural network. That is learning rate, batch size, epochs, optimizer parameters. As often done for neural network models, these can be hand-tuned to improve the performance on a specific task. **Learning Rate:** The learning rate controls the parameter update size per gradient descent. This is often tuned. In practice, for DO 0.001 works well. **Batch Size:** Batch size controls the number of training examples used at one time during a single back-propagation bass. The error gradient is calculated as a sum of the gradients for each training example in the Batch. A small batch size generally improves learning as the signal to learn can be clearer. However, training can be very slow with large training set and therefore should be tuned appropriately. For experiments with training set size below 5000, we have found a batch size below 10 to be appropriate. **Epochs:** Epochs defines the number of cycles through a training set are performed during learning. Concretely, how many times a single training example is used during the training phase. This can significantly improve DO as the model optimisation process is gradient based. Therefore, the training set may contain sufficient information to learn but requires multiple exposures to apply sufficiently updates to the parameters to find the best model parameters. However, too many epochs can results in over-fitting of the training and therefore can require tuning. In practice, we find that the population size has a more significant impact than Epochs, and 20-50 epochs works well. **Optimizer Parameters:** Learning rate forms the primary parameter for optimisation methods as it defined the size of the gradient step to make at each iteration. However, more advanced methods such as momentum and Adam optimizer incur further parameters. We have found that using a momentum rate of 0.5, in both Momentum and Adam optimizer to work well **Training Set Size:** When DO is run in incremental mode, the training set size is a single update to model. However, the transition timing provides an effective training set size - the number of solutions required to induce a good representation of the solution

In Do, the number of solutions (training data points) is not limited as they are generated. However, the performance of DO with regard to scalability requires tuning of the training set size. The training set size defines the number of solutions that are used during training. The optimal size is only that incurs minimal cost to generate (number of function evaluations) yet provides sufficient data to learn the structural information. This parameter can be tuned by measuring the performance of the model. Of course, the most accurate measurement to decide the performance is to use the model on the task. However, this is not possible for DO as we do not know ‘a priori’ what features of variation are most suitable. We have found the reconstruction error (of the corrupted input if denoising is used) provides a good approximation of the model performance.

4.3.2 Solution Optimisation

Aim of section - demonstrate the optimisation cycle, given that learn and training are understood (or can be from other sections) - all about how: - local search is performed at the latent representations - How this is decoded back to the solution level so that movement in latent space can be evaluated - How this provides a signal to learn relationships that contribute to the solutions fitness.

The motivation of DO is to exploit the ability to perform MIV at multiple scales of representation and that MIV is capable of following fitness gradients that MIG only cannot. Therefore, the model is encouraged to learn a representation that lends itself naturally to MIV. Therefore, MIG can become redundant in this scenario as the latent distribution can be to complicated to sample.

However, it is nevertheless still useful. While MIG may not provide an advantage for moving in the search space compared to MIV, it can allow for concentrating the search in areas of greater interest. For example, take a landscape that traps solutions in areas that are not meaningful [1] – caused by noise, poor variation or constraints on the landscape.

4.3.2.1 Model Exploitation

In rA-g, at the start of each solution initialisation the solution was reset using a random uniform distribution. This allowed for exploration of the search space. However, for DO in batch mode, the distribution is of solutions is optimised before updating the model. Therefore, a solution in the population is updated, rather than being reinitialised. This is inline with the operation of EDAs and LTGA. However, a unique aspect of DO, as we discuss in more detail in section 4.5 is the ability explicitly maintain a diverse population - due to the encoder transforming the solution to a representation that can be used to initialise the location in the new neighbourhood for local search to perform. A solution in the population is therefore only directly replaced by a solution that was an update from the encoded solution. This maintains the cover of basins of attraction in the solution space.

As discussed in Chapter 2 Exploiting the compressed representation can be done in one of two ways. One method is to use a probabilistic model and sample it to generate new solutions. We call this approach model-informed generation, as the model is used to generate new solutions. The other method is to use a deterministic model and to extract higher-order variations to update solutions. We call this method model-informed variation, as the model is used to

For instance, 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”). We then desire a model that represents this relationship. In MIG, the model generates 00 and 11 more often than 01 or 10. In MIV, the model represents 00 and 11 as neighbours even though they are not neighbours in original space.

Interpreter Function Output of the decoder network is in continuous space. Interpreter function transforms a decoded output into a solution. The interpreter function is dependent on the type of problem. In this thesis, the main focus is on binary optimisation

[1] Not meaningful referring to variable combinations that do not contribute to the solutions quality. Therefore exploiting this information is unlikely to improve a solution

problems. However, unlike other MBOAS, DO is not limited to a single solution representation. As we explore in Chapter 7. DO can be applied to continuous solutions and discrete problems with greater than two variables.

Unit function

Model-Informed Generation A restart hill-climber works be randomly sampling the solution space to initialise a solution. This is beenficiial as it allows the algorithm to explore the search space. DO can perform the same process before MIV.

By encoding the solutions to intilise the hidden layer ensures automatic niching.

MIG is the process of generating an entire candidate solution using the model. This is achieved by using the autoencoder model as a generative model. In general, given a training set with distribution p(xr—x) we wish to maximize the likelihood of the estimate. We can therefore interpret the latent representation as providing a conditional distribution pdec(xr—h). Xr is conditionally independent given h. Therefore, each unit of xr can be sampled independently to generate a complete solution. To generate Xr, we therefore need h. DO does not use a VAE and therefore the distribution for h is unknown and cannot not be sampled directly in a meaningful way. Never-the-less this can be overcome using an empirical distribution or by encoding a solution to parameterise the latent distribution

Empirical The distribution of h can be empirically approximated by encoding all the solutions (or a random sample of solutions) that were used for training the model. This provides an empirical distribution of the activation response. H is then generated assigning each hidden unit with a previous activation response chosen at random. Encoding H is generated by encoding a solution.

Encode a solution To generate a new solution, first a solution is encoded to encoder the parameters for the latent distribution (the means to sample from)

The algorithm is as follows: For P in Population.Size: Solution = Population.Solution[P]
HiddenRepresentation = Encode(Solution) SampleLatentRepresentation = Sample(HiddenRepresentati
DecodededSolution = Decode(SampleLatentRepresentation) NewSolution = Interpret(DecodedSolution
NewSolutionFitness = Fitness(NewSolution) If NewSolutionFitness ¿ Solution.Fitness:
Replace(NewSolution, Solution)

This allows for an implicit diversity maintanece as new samples are parameterised by samples that it is in direct competition with.

However, the autoencoder does no represent an latent space that allows for each hidden neuron to be independently sample. VAE's are more appropriate here. Never the less, we include to demonstrate the performance difference between MIG and MIV when using the autoencoder model. Further, the method of MIG is inline with other EDA methods that use a neural network, and therefore provide a comparitive method.

DO can use MIG to initialise a solution before then applying MIV

Generating that biased by the current solution - therefore providing a method for diversity maintanence, keeping in differnet, interesting and potential useful areas of the search space.

Model-Informed Variation

Figure 4.7 - Weights of an single layer autoencoder model that has been trained using a distribution of locally optimal solutions to the MC problem

Figure 4.8 - Illustration of MIV - demonstrating how a local change in the encoded representation transforms to a high-order change (simultaneous change to multiple variables) at the solution level. The encoder network calculates a hidden representation that corresponds to the current solutions location in the search space. The decoder network transforms the movements in the latent space back to the original solution space. This allows for the the fitness of a solution, and thus movement in latent space, to be evaluated.

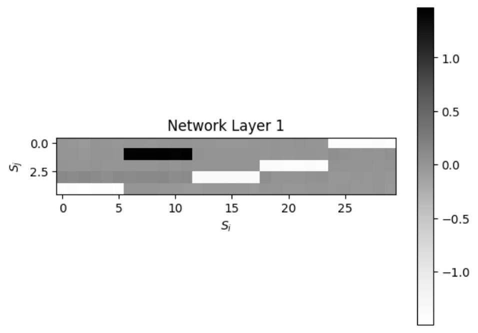
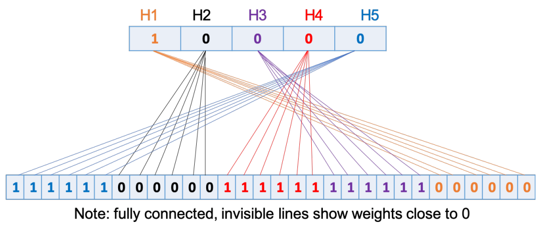


FIGURE 4.7: The weights learnt by the Autoencoder on the MC problem, size 30, with k=5. The Auto-encoder represents 5 separate building-blocks of size 6.

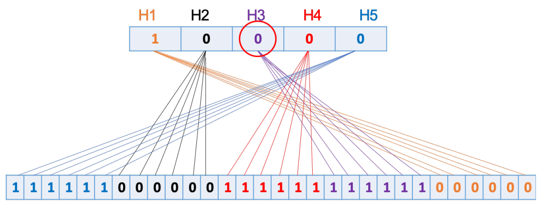
In this section, the details for performing a partial change to an individual solution is presented. The partial change is determined by a local change at the hidden representation. As determined in Section [ref], a change to an individual hidden variation was sufficient to navigate the rugged landscape of the MC problem. However, strong assumptions were made about the relationships captured by the model. Specifically, the autoencoder learnt a lossless compression of the solution space (providing an almost perfect reconstruction of a solution), the compression ratio was close to optimal and interpreting the hidden representation as a binarised code was sufficient for the problem.

Therefore, the method for MIV has been developed to increase the robustness for searching in the hidden representation. Methods for model-informed variation

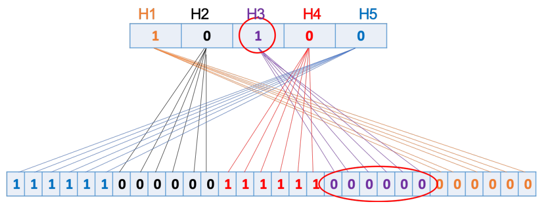
a) Representation of the solution - Encoding the current solution - accounting for neutral variations (overcome when encoding) b) Change to the latent representation - Direct


H1
H2
H3
H4
H5
Note: fully connected, invisible lines show weights close to 0

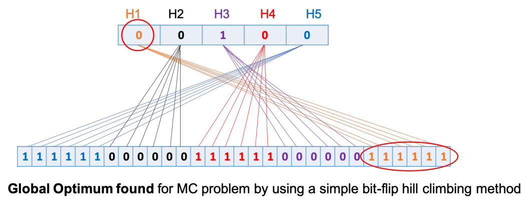
(a) Encoding of a solution from the population (a local optimum in the original solution space.)



(b) The third hidden neuron (H3), responds to 4th building-block.



(c) As the weights are tied, a change to (H3) decodes to a change to the 4th building-block. Providing a simultaneous change to all variables in the building-block, enabling search to escape the local optimum and improve the quality of the solution


H1
H2
H3
H4
H5
Global Optimum found for MC problem by using a simple bit-flip hill climbing method

(d) The process is repeated, where a local variation applied to H1 decodes to a change to the 5th building-block

FIGURE 4.8: Illustration of how DO performs MIV and the subsequent effect it has on a solution. The network is fully connected, coloured lines indicates a neurons dominate weights - as seen in Figure Figure 4.7.

change to the latent representation - Empirical change to the latent representation - Encode a change at the solution representation c) transforming the latent representation to the solution representation - Direct decoding - Partial variation

The model-informed variation method (MIV) used by DO can be intuitively described as performing 'local search at the latent representation'. The algorithm for MIV, that updates each solution in the population, is presented in Algorithm 5

At first, the model contains no meaningful information about the problem structure. Therefore, local search is limited to the solution representation only (MIVlayers = (0)). The change to a solution (ΔS) is set as a random single-variable variation to $S[i]$. The fitness effect due to the change is then calculated (ΔF). If the fitness has been improved, then change to the solution is retained, otherwise it is rejected. This is performed until no single-variable variation provides a fitness improvement for the solution.

After training, the autoencoder model is used to update a solution. The autoencoder model transforms the neighbourhood of a solution. A point in that neighbourhood is represented by the latent variables. ΔH represents the local variation made in this new neighbourhood. A simple method to move in the latent representation is to make a random variation to a single hidden unit. Empirically, this can work well but generally requires the autoencoder model to learn a good quality representation.

In DO, the representation induced will affect the performance of MIV. A representation that has a poor separation of building-blocks, or poor-representation of the relationship, will not allow MIV to perform well. Therefore, it is important that a good representation, and therefore good feature extraction, is performed. Further work will be required to explore the synergy between inducing a suitable representation and searching within this space to improve the robustness of the algorithms performance.

MIV is the process of making partial movements to a solution. We therefore need to calculate Δx where $X = [x_1, x_2, .., x_{n-1}, x_n]$ using the model. This is achieved by perturbing h by Δh and calculating the response Δx. The Δx then provides the information of how to move the solution.

Perturbing h: As at the solution, we do not know what change to make, where and where to make. To make things more complex, at the hidden representation, the values are continuous.

The robustness of the method depends on how well the model represents the features of variation in the data.

Requirements for partial movements:

1. A direction: this is represented by the learnt parameters.
2. A magnitude: this is defined by Δh.

Thus, the weights of the autoencoder define the direction of movement in the solution space that can be made. Δh defines the magnitude of the movement in a direction. To provide an intuition about how to decide what Δh, we apply MIV to the solution level as this definition is general (it is applicable to all layers of representation). We can create an AE with a hidden layer representation that has learnt the identity function (I). To search in this space, generally we use a bit flip algorithm. This is because we want to use a variation operator that does not prematurely restrict the degrees of freedom of variation. Therefore Δx = 0,0,...,1,..,0,0 and thus dH = 0,0,. . . ,1,. . . ,0,0,0 when W = I. Therefore, a bit-flip variation corresponds to a change to a single hidden neuron at the hidden representation.

The idea of direction and magnitude - basis vectors applied to a solution and the hidden representation.

Methods for calculation dX: *Size of dH* Assuming the hidden node represents a basis vector of variation, then the hidden node value represents the vector component of the basis vector. A change the vector component causes a movement along the basis vector. The distance and in which direction therefore corresponds the change in the activation value, $\Delta h = h - h'$. The methods we explore for calculating Δh are as follows:

Empirically Method: At transition, an empirical distribution of the hidden layer can be generated by encoding all solutions in the population used for training. This provides an empirical distribution of the activation values. Δh is calculated by random selecting a single hidden unit and then randomly sampling the empirical distribution to obtain a value for the selected hidden unit. The difference between the encoded value and the randomly selected value provides $\Delta h = h_i - h_i^e mpirical$

Flip: From the preliminary studies using rA-G, the induced representation has a saturated response - the hidden neurons were near the minimum or maximum activation values. Using tied weight encourages this

That is assuming that hidden units are encouraged to saturate towards the minimum or maximum activation values. It is unlikely that a dh between within the saturation region of a activation (between 0.95 and 1.0 for example) will contain and unique solutions). In this case, we can therefore just perform a flip of the activation value to calculate dH. This removes the requirement to store and sample the empirical distribution. By encourage a representation that saturates the activation's will increase the performance.

Assign: In addition to flipping the value of a bit, a more robust variation to substitute in the maximum of minimum activation value. This has connection with sampling the hidden layer rather than using the actual value. This alleviates the problem, when activation values are in-between activation's and need to be increase. Here a flip variation will not succeed. This is a small adaptation to flip variation and is expected to make only a small change the efficacy to the performance of DO.

Encode

The encoder network is used to inform which hidden units to change, providing a more robust mechanism for moving in the latent representation. First, $\Delta H = E(S + \Delta S) - E(S)$, where ΔS is a single-variable variation. ΔH represents the sensitivity of the latent units to the change to $S[i]$. As the weights are tied, ΔH provides an approximation to the direction and magnitude each hidden unit moves in-order to cause the variable $S[i]$ to change value. It has been found that directly using ΔH is not always sufficient. Therefore, a random threshold between the average and maximum magnitudes in ΔH is used to filter out hidden units that do not contribute to the change in S_i. For the hidden units that show a sensitivity above the threshold value, the change made to these hidden units is set such that in $H' = H + \Delta H$ the units in H' are at the minimum or maximum activation value.

This difference is not sufficient on its own to determine the local variation to make H. DO uses dropout. Therefore in order to have a low reconstruction error, the model must be able to generate the values to the missing variables using information from other variables. This is useful for encouraging the model to learn correlation between variables (as this improves the ability to generate values to variables from other variables).

However, it also enables a robust representation. Specifically, a small variation to the input is reconstructed back to the original input. Therefore, applying dH directly to H is likely to cause a negligible changes to the decoded output (especially as a unit step function constructs a solution from the reconstruction of the decoder).

This method overcomes the case when hidden units share the same, or similar, transformation. For instance, if there exist multiple hidden units that have the same connection weights with the input, the decoded signal will be suppressed by the other hidden units that were not changed (the other hidden units will still want the solution variables to remain as they were). However, all these hidden units will show a strong association with the same variable(s), and thus have a similar response to a change in a solution variable. By approximating which hidden units are responsible for the value of a single input unit, a variation to all these units provides a more robust method to making a sufficient change at H that will cause a change at S.

Algorithm 5: DO - Solution update via MIV

```
for Solution in Population do
    for L in MIVlayers do
        while Solution can be improved do
            $\Delta S = 0$ ;      // Initialise variation to apply to each variable in Solution
            $i = U(1, N)$ ;                              // Select a variable to change
            $\Delta S[i] = 1-\text{Solution}[i]$
            if $L > 0$ then
                // Calculate latent variation from change at solution representation
                   using encoder
                $H = \text{Encode(Solution,L)}$
                $H^s = \text{Encode}(\text{Solution}+\Delta S\text{,L})$
                $\Delta H = H^s - H$ ;                                    // Change at H
                $\Delta H\_mag = |\Delta H_i| \colon \Delta H_i \in \Delta H$
                $a = \frac{1}{N_l}\sum_{i=1}^{N_l} \Delta H\_mag_i$ ;
                $z = \max(\Delta H\_mag)$ ;
                $T = a + (z - a) \times U(0, 1)$ ;                          // Threshold
                $\Delta H = \begin{cases} u(h_i^s) - h_i & \Delta H\_mag_i > T \\ 0 & otherwise \end{cases}$
                $H' = H + \Delta H$
                // Calculate solution variation from change at latent representation
                   using decoder
                $S_r = \text{u}(\text{Decode}(H))$
                $S'_r = \text{u}(\text{Decode}(H'))$
                $\Delta S = S'_r - S'$
            $S' = \text{Solution}+\Delta S$
            $\Delta F = F(S') - F(\text{Solution})$
            if $\Delta F > 0$ then
                // Fitness improvement
                $\text{Solution} = S'$
```

Calculating Δx

A new model-informed variant, S', is constructed from S using $S' = S + (S'_r - S_r)$, where S'_r is the solution reconstruction of H' and S_r is the original solution reconstruction. If the change to S improves the fitness, the change is accepted, otherwise it is rejected. The process of MIV is repeatedly performed until no variation provides a fitness improvement for the solution.

Once a Δh has been calculated, the corresponding change to ΔX can be calculated using the decoder network. The decoder function is used to generate $Xr = D(h)$ and $Xr' = D(h + \Delta h)$. The change made to the solution ΔX is then calculated as $\Delta X = X_r - X - r'$.

The reason for calculating ΔX as the difference from the reconstruction of the solution and not the solution itself is because it is unlikely that the autoencoder model will provide a perfect reconstruction of the solution. The model extracts the salient features from the data-set and higher-order features, noise, or rare features are likely to not be well reconstructed. Therefore, in calculating ΔX with respect to the original solution could cause changes to other variables, not because the model captured this relationship, but because the relationships was not represented by the model. calculating ΔX with respect to the solution reconstruction improves the calculated variation to make to the solution.

Interpreting dX:

ΔX defines the direction and magnitude of variation to make to a solution. The output of the Decoder network is in continuous space and therefore needs interpretation to apply to a solution in discrete space.

Direct Interpretation The decoded output is used for solution reconstruction via the interpreter function i.e., $S' = X'_r$

Magnitude change Interpretation If ΔX_i, where i corresponds to an individual unit of ΔX, has a magnitude greater than a specified value (i.e., a specific or random distance between the activation extremes). Then the model is making a clear decision that v-I should be changed in the direction of dx_I.

Difference between Solution Interpretation: The solution interpretation relaxes the size of ΔX to create a variation. This is achieved by measuring ΔX as I(ΔX_r) - I($\Delta X'_r$) where I is an interpretation function of the decoded output that is unique to the problem. In the case of a binary problem this can be a unit threshold hold function.

The vector px represents the partial change. The variables to change are represented by assignment in px that are not equal to 0, the value to assign to variables to be change are represented by the values corresponding values of px.

Overcoming Neutrality at the latent representation: Overcoming the problem when many neurons in the hidden layer respond to the same feature. Therefore, in order to make a partial change at the solution level, the majority of the hidden neurons that respond to that feature must also change. Otherwise the output of the decoder

When using a L1 regularisation and non-linear output activation's the model that maximises the reconstruction and minimises the regularisation term is one that uses many neurons to do the same things. This is because the regularisation cost and reconstruction cost are non-linearly proportional. As the reconstruction improves, reaching saturation points, the weights must improve significantly to make a small increase in the output value. However, this incurs a larger cost in the regularisation term and is therefore discouraged. One method to overcome this is to use a linear output activation. However, this does not overcome the issue with at deeper layers due to the non-linear hidden units.

Instead, the method to overcome neutrality is, when a change at the latent representation corresponds to no change at the solution level, the change to the latent representation is kept, memorised and applied to the next iteration step. This allows for an accumulation of neurons to change. When a change does occur at the solution representation, the accumulated latent changes, previously accepted due to neutral effects, are discarded and the process repeats.

We are yet to fully understand the importance between the differences, but our experiments suggest model-informed variation is significantly more important than model-informed generation.

Multi-level Search

The MIV algorithm presented here allows for a generalised method for searching at multiple neighbourhoods defined by the model (searching at multiple latent representations). The method for traversing the autoencoder model can be bottom-up (MIVlayers = (0, 1, . . . D), top-down (MIVlayers = (D, D-1, . . . 0) or bottleneck only (MIVlayers = (D)).

Repair Searching in the original solution level can provide repair Hypothesis as a useful to included - memetic algorithms, local search in EDAs Restricted to a single level of organisation

We find that it improves the robustness of DO. The model can cause disruption to some building-blocks in a solution, in favour of

DO constructs and maintains a multi-level representation, enabling repeated search at different levels of representation - a type of repair at the lower level representations, or continuing search in a more adapt neighbourhood (induced by learning higher-order features)

By maintaining the solution and only accepting solutions, provides a repair operator the next time the solution is update (if needed)

This can be done by traversing the model bottom-up or top-down.

Bottom up search A solution is updated by search each representation level from the original solution level to the deepest, training layer in the autoencoder model.

MIG provides a method of searching in a new space of the neighbourhood, initialized by the solution to update. Preliminary experiments did not show a measurable performance improvement compared to MIV alone. However, this remains a promising area of research as it combines the functionality of MIV and MIG into a single model space.

bottom-up/top-down and a type of repair has connections with other structures such as critical back-bone Kilby et al. (2005); Prugel-Bennett (2007). A backbone of an optimisation problem is described as the subset of variable assignments that puts the solution in the correct basin of attraction to find the global optimum. Once in the correct basin of attraction, search using simply methods (such as hill-climbing or lower order variation operators in thh case of DO) can be used to refine the search to find superior solutions. It is therefore hypothesised that search at multiple levels of representation can significantly improve the performance of search procedure. High level organisation can provide a method for setting the critical variables that maneuvers the solutions to a good basin of attraction. Lower order search, such as those in the lower levels of a network can then be used to refine the search to find a superior solution, search that a higher-order variation is not able to do - a type of repair using lower order variation.

4.3.3 Replacement

Discus how this maintains use-full information in the dataset, that may not have been exploited yet

A limitation by EDAs when performing MIG is during replacement. After generating a solution, a decision must be made into how to incorporate this into the new generation training set. Early methods simply discarded half the population and generated solutions to replace these solutions. However, much research has identified this to be an ineffective method as one can quickly loss diversity of the population and override solutions that may have had proportionally lower fitness but were nevertheless containing important structural information. Multiple techniques have been developed to improve the diversity maintenance of a training set - a research topic of interest in the Evolutionary Computing community

The autoencoder model used by DO allows for in explicit diversity maintenance due to the encoder model.

Individual competition with the solution used prior to the solution optimisation steps. If the outcome of MIG or MIV has improved the solution. The new solution replaces the existing solution in the population.

Update Solution - the advantage of a distribution of solutions is that we can maintain solutions in interesting basins.

The solution optimisation cycle produces locally optimal solutions as guided by model-informed variation. Specifically, a candidate solution X is initialised from a random uniform distribution. A random variation is applied to the candidate solution, forming X', if the variation has caused a beneficial fitness change, or no change to the fitness, the variation is kept, $X = X'$, otherwise the variation is rejected. This procedure is repeated until no further improvements. All solutions in the distribution are optimised before the model is updated.

Evidence of MIV - The higher-level search has coarse grained basin of attraction or a smoother landscape at that level of organisation.

A decoded variation made to the hidden layer causes a change to the solution level that exploits the learnt problem structure. Concretely, module-substitutions are constructed by performing bit-substitutions to the hidden layer and decoding to the solution level.

As DO uses an unsupervised learning algorithm, for it to learn a meaningful representation the training data must contain information about the hidden problem structure in its natural form. This structure becomes apparent when applying a hill-climbing algorithm to a solution because it ensures that it contains combinations of variables that provide meaningful fitness contributions. Initially, the AE model will have no meaningful knowledge of the problem structure and therefore

Once a transition has occurred, DO does not require knowledge of which operator has been initially used, it simply learns and applies its own higher-order variation operators.

4.3.4 Transitions

Transition is a term used to define the moment when the bottleneck layer has model the training data sufficiently well that this information can be exploited.

Determining when to transition requires an understanding of when to exploit information from the model. That is, when the model has learnt a good representation of the training data. As we have discussed, the definition of a good representation is vague and we do not discuss the precise interpretation here as it is specific to MIG or MIV, the regularisation used (what representation DO is encouraging). As used for automating the termination of training, we use a cross-validation error provides a useful measure to inform us of how good the model captures the true relationships in the data.

After the network has learnt a good meaningful representation at hidden layer n the following changes occur to DO, which we term a transition.

1. An additional hidden layer H_{n+1} is added to the AE. Previous learnt weights are retained and training updates all weight (W_1 to W_{n+1}).
2. The hidden layer used for generating model-informed variation is changed from H_{n-1} to H_n. Initialisation of a candidate solution is generated from H_n.

Item 1 is analogous to the approach introduced by Hinton Hinton and Salakhutdinov (2006) for training DNNs. The layer-wise procedure is important for learning a tractable representation at each hidden layer. The multi-layer network is trained on solutions developed using variation decoded from the layer below the current network depth. This is a significant requirement as DO learns from its own dynamics **?**. There may be many possible mappings in which the problem structure can be represented. Thus deeper layers are not only a representation of higher-order features present in the problem, but are reliant on how the higher-order features have been learnt and exploited, which, in-turn, is determined by the shallower layers. Therefore if the shallower layers do not contain a meaningful representation, then attempting to train or perform variation generated from deeper layers will be ineffective as we prove in Section **??**. Item 2 is a layer-wise procedure for generating candidate solutions. The method of generating variation to the solution is the same at any hidden layer. Simply, only the hidden layer where bit-substitutions are performed and decoded from has been changed from H_{n-1} to H_n (to a deeper hidden layer).

This transition procedure is performed recursively until the maximum depth of the autoencoder is reached at which Item 1 is not performed. Figure**??** provides an illustration of the network architecture before (a) and after (b) transition. Like the learning rate, the timing of transition impacts the balance between exploration and exploitation of the search space. Once transitioned not only does the model provide information on how to adapt the applied variation but the solution optimisation cycle provides feedback to the model optimiser. Specifically, correctly learnt features will cause beneficial changes to a solution during optimisation, and therefore will be repeatedly accepted during the solution optimisation cycle and thus repeatedly presented to the model during training, reinforcing the learnt correlations. In contrast, incorrectly learnt features will cause deleterious fitness changes and therefore will not be accepted and thus not present in the training data.

Figure **??**.c presents an illustrative example of a fitness cross-section for a problem containing deep structure where local optima are present at each level of the hierarchy. The cross-section illustrates the fitness landscape apparent to the variation operator. Variation at the solution level X, refers to changes of individual bits, at the 1st hidden layer, H_1, changes of module solutions and at the 2nd hidden layer, H_2, changes in meta-module solutions (combination of modules). At the solution level, X, there exists many local optima that trap the bit substitution operator. DO learns the features present in these locally optimal solutions and exploits this information such that searching at a hidden layer causes a higher-order variation that can escape these ‘shallow’ local optima. However, the adapted search operator can itself get trapped at relative local optimum we term ‘deep’ local optima. A further layer is then required to learn how to escape these deep local optima. This recursive adaption of the search operator to higher-orders of organisation reduces the dimensionality of the search space and causes an effective coarse graining of the fitness landscape at each depth such that at the deepest level the fitness landscape peaks are the global optimum and easily reachable using a hill-climbing algorithm.

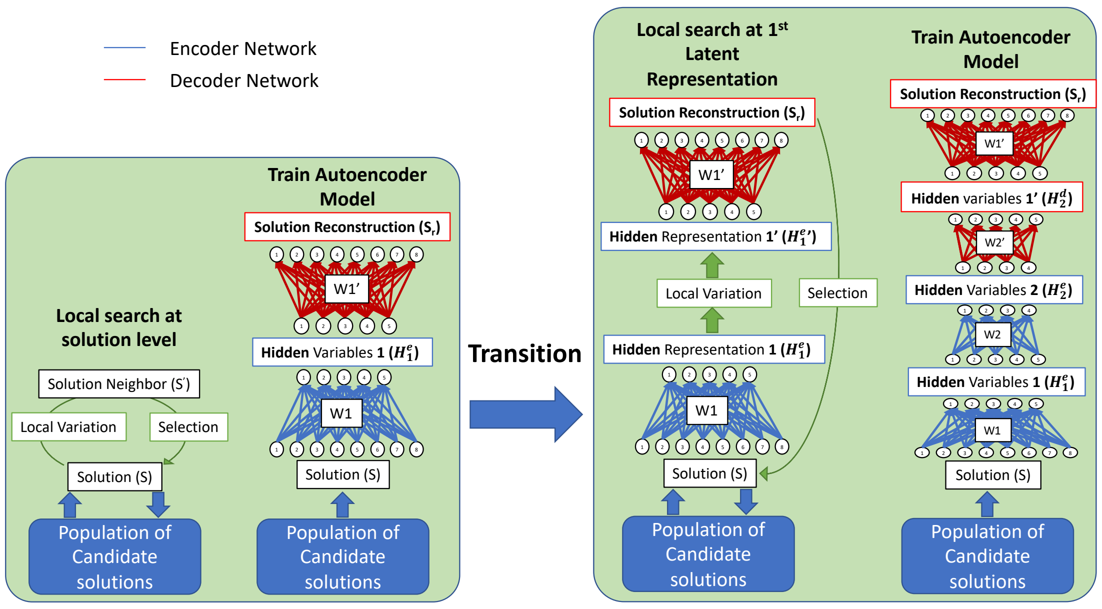

Encoder Network
Decoder Network
Local search at solution level
Solution Neighbor (S')
Local Variation
Selection
Solution (S)
Population of Candidate solutions
Train Autoencoder Model
Solution Reconstruction (S_r)
W1'
Hidden Variables 1 (H_1^e)
W1
Solution (S)
Population of Candidate solutions
Transition
Local search at 1st Latent Representation
Solution Reconstruction (S_r)
W1'
Hidden Representation 1' (H_1^e')
Local Variation
Selection
Hidden Representation 1 (H_1^e)
W1
Solution (S)
Population of Candidate solutions
Train Autoencoder Model
Solution Reconstruction (S_r)
W1'
Hidden variables 1' (H_2^d)
W2'
Hidden Variables 2 (H_2^e)
W2
Hidden Variables 1 (H_1^e)
W1
Solution (S)
Population of Candidate solutions

FIGURE 4.9: Illustration of the first transition occurring in DO. An autoencoder model with a single hidden layer is trained using the distribution of solutions found by variation and selection at the solution level representation. Initially, variation is a single variable variation. After training, transition occurs. Each solution is then updated using MIV at the 1st hidden layer. Then, a layer is added to the autoencoder model and trained. Training updates all model parameters using the distribution of solutions.

4.3.4.1 Layerwise Construction

The recursive application of a transition constructs a deep representation of a solution.

The autoencoder model is constructed using a layer-wise construction procedure. At initialisation, the autoencoder has a single layer, $L = 1$. The model is then trained using a population of solutions. The first transition then occurs. Specifically, Model-Informed Variation (MIV) is applied using a higher-order representation of the solution, represented by the latent space H_1. MIV is repeatedly applied to a solution until no further improvements can be made. This is performed on all solutions in the population, producing a distribution of solutions that are locally optimal relative to the latent representation. A new hidden layer is then added to the autoencoder model ($L = L+1$).

The autoencoder is then trained using the new distribution of solutions and updates all layers of the model. The second transition then occurs, where solutions are updated by MIV using the second layer latent representation H_2. This is repeated, constructing a deep autoencoder model. Figure 4.9 presents an illustration of the search and training before and after a transition.

Layer-wise Building Layer wise building is the main procedure used for DO for generating a deep representation of the variation.

Layer-wise Stacking Stacking is the general term used to describe the process famously introduced by Hinton and Hinton and Salakhutdinov (2006) of greedy layer wise training. The process involves multiple single layered auto-encoders and assigned a depth from 1 to maximum depth. The shallowest autoencoder learns a reconstruction of the input. Once trained, based on some empirical determination, the Depth 2 autoencoder is trained to learn a reconstruction of the hidden activation response of depth 1 autoencoder. Thus, a solution is input into the depth 1 autoencoder and the activation responses are used as the input the depth 2 autoencoder. This is recursively applied such that depth 3 autoencoder learns the reconstruction of the depth 2 activation response.

Never-the-less, stacking provides an interesting property for optimisation. It explicitly separates the search neighbourhood and provides a method to move through each space. This property becomes apparent when we compare the behaviour of layer wise building as we detail next. The representation learnt at the bottleneck layer is only in reference to how to vary in the space of the layer below, where as in layer-wise building, the bottleneck layer is in reference with the input layer.

Note that state-of-the-art results regarding multi-layer autoencoder models rely on a stacking of shallow models. It is only in the task of classification; they are joined to create a feed-forward deep network. For generative task, they generally remain as stacked shallow auto-encoders.

4.3.4.2 Number of Solutions

Transition requires determining if the trained model has a good representation of the data. This determination can be performed by a practitioner who manual sets the size of the training set. This will require repeated attempts to find an adequately sized training set size and manual interpretation. This method is generally used in MBOAs and is referred to as the population size.

For machine learning we can utilise the cross-validation method to help us understand the performance of the model and therefore guide us if more data points are required, or the model has overfit to a poor representation.

An empirical algorithm was developed in order to automate transitions:

Population = [] empty population Transition = False CV$_{E}rror = LargeNumberWhilenottranistioned :$
$Population = addNewSolutions(N)addNsolutionstothepopulationOptimiseSolutions(N)UseMIVtooptimis$
$ReconstructionError(Population[N])reconstructionerrorfornewsolutionsonlyIfNew_{C}V_{e}rror <$
$CV_{E}rror : CVerrorwentdownwithlastbatchsocontinuetrainingCV_{E}rror = New_{C}V_{E}rrorTrainModel(Popula$
$CV_{e}rrorsaturated, thereforetimetotransitionTrainModel(Population)UpdatemodelwithnewdatafirstPerf$
$True$

Initially the training set is empty. DO adds N new solutions to its distribution of solutions and performs MIV. These solutions are first interpreted as the validation data and its reconstruction error determined before it has been used to update the model, thus acting like a cross-validation set. If the error is lower than the previous validation set error, then this indicates that previous training improves the generalisation of the model. That is, more data improved the model's representation of the true relationships in the data. Therefore, it is good to keep increasing the size of the training dataset. Therefore, DO adds the validation set to the training set (the training set uses all old solutions and newly generated solutions) and then proceeds to update the model. If the validation error is greater, then this indicates that adding more data to the training set has either not improved the mode or made a negligible difference. Therefore, it is unlikely that adding more data will improve the model's performance and thus DO transitions to exploit the information it has captured.

We provide empirical results that show this method worked empirically in Chapter 6.

4.3.5 Additional Implementations

The reason is due to the nature of DO – that its learns a representation of its own dynamics. Therefore, if the model has learnt an model that underfits the data, then all we loose is efficiency to exploit information that was available in the training set. However, we can still explore the search space, the model has not restricted movement

4.3.5.1 Depth-Limited DO

To access the importance of a multi-level representation, it is important to understand differences between limited variations of DO. DO encompasses many aspects of a deep learning model, with design decision based to maximise the utilisation of a deep model. The focus of this research project is to understand how inducing multi-level representation can enhance optimisation. Therefore, to empirically show that a deep representation can overcome problem challenges that a single layer representation (shallow) cannot, we can control the depth limit of to a specified depth, with notation DO$_{\mathrm{d}}$ where d represents the maximum depth of the bottleneck in the AE.

Limited DO refers an instance of the DO algorithm using an autoencoder model with a maximum number of layers. Therefore, at transition, when the autoenocder model has reached the maximum number of layers, instead of adding a new layer to the model, the deepest layer is reinitialised at training. Search continues to use to deepest representation. The algorithm then continues in its normal operation.

When DO reaches the limit, the deepest layer can either be maintained and training continue, or the deepest layer is reinitialised. We have found in practice that revitalising the deepest layer enables the network to escape attractors in model space that trap it from learning higher-order features. For instance, a representation that captures features that become positively correlated at the next has the same reconstruction error as when they varied independently.

As generally employed by the state-of-the-art MBOAs, the model is reconstructed at each generation. That is, the model learnt is discard after performing variation and a new one, initialised from random is constructed.

This presents an opportunity to remove relationships that are no longer necessary yet restrict learning new relationships due to trapping the weight space in a local optimum. For example, given a model that has learn a perfect representation of the variation structure. This is exploited to generate a new training set. This training set will contain a combination of the features found by the previous. Thus, while there is a reduced dimensionality of the training date (features are combined) the model will not have a pressure to represent this. Separating high-order features into its components still present a low reconstruction error. Therefore, method to overcome this is to reinitialise the weights. Now the model will learn only the higher-order features and not its composition

Limited DO enables a method for understanding how a deep representation can enhance evolutionary computation and not just to escape local optimum in the parameter space of the model.

4.3.5.2 End-to-End version

4.3.6 Interpreting What DO has Learnt

Methods for understanding what the autoencoder model has learnt Generally performed by visualisation of the model to understand what it has learnt - how it computes the function - Weights (filters) - Plotting inputs in new coordinate space - similar points closer to each other in new dimensional space. - Interpolation at latent layers - Decoding stimulus to latent space.

Can be used in BBO to uncover structure of the problem instance and possibly problem class

4.4 Summary

Updating the model in DO using candidate solutions as the training data is no different to the training performed in other machine learning tasks. The same goal is desired – obtaining a model that captures the true data relationships of the training data so that we can perform some computation using the relationships (be that classification or generation). In DO, this can be manually tuned to suit the optimisation problem. However, we have introduced methods that enable an automation that decides how long the training phase takes, how much data to use in the training set and thus when to perform a transition. Automation and removal of these hyperparameters is an important step to make DO a viable method for practitioners. We find that this automated technique empirically works well at this early research phase. The sensitive of manually defining these hyperparameters appears relatively low, and this is encouraging as mistakes in automating the decisions are therefore unlikely to cause catastrophic failure, and rather just efficiency. We hypothesis that this is primarily due to DO being a process that learns from its own dynamics. If we tune parameters such that we never over restrict the degrees of freedom, then at worst we will be less efficient rather than not being able to explore. The idea that if in doubt aim for simplicity. DO can then slowly compress the space when the signal becomes clearer later in the search.

DO provides an algorithm that provides a single algorithm that connects both EDAs and MSS operators into a single algorithm i.e., both MIG and MIV can be used to produce new solutions. Therefore DO provides an optimisation method for incorporating the methods of both EDAs and MSS in a single algorithm.

It is hypothesised that the Deep Optimisation algorithm is not limited to the autoencoder architecture. We expect the concept to be applicable to other architectures that have been widely used in other domains, will also by suitable in the context of Deep Optimisation. For instance, Convolution Neural Networks (CNN) for transnational invariant features. Although, other architectures are not explored in this thesis and instead concentrate on providing the foundation the connection.

State-of-the art machine learning methods – bring advanced machine learning tools to the framework of MBOA.

Questions that arise from the development of DO, and previously indicated in Section []

1. Does their exist problem structure that requires a deep neural network to exploit
2. How does layer-wise model-building vs end-to-end modeling building compare.
3. What are the types of problem challenges a neural network model can overcome that other models, used in other SOTA algorithms cannot.

4. What are the types of problem challenges that local search at new representation can overcome that crossover or random recombination cannot.
5. What are the types of problem challenges that explicit search at multiple levels of representation can provide

4.5 MBOA Functional Comparison

Goal of section:

Discuss weaknesses highlighted by using this comparison - Comparision made from the perspective of an individual solution - new model - doesnt remeber the dimensional compression performed (except p3) - Representation space is always the original space (not a non linear transformation) - Type of feature extraction used (bivariate, multivariate statistics.)

What makes DO novel from state-of-the-art MBOAs is the use of a deep neural network to represent multiple scales of organisation. A further differentiation, also explored in this thesis, is the method used to exploit this information to search the solution space of an optimisation problem, namely the method of multi-scale search. This research project focuses on developing our understanding for how a deep neural network model can be used to provide state-of-the-art performance. Specifically, we answer the following three questions:

1. How to induce a higher-order representation of a solution, and what information this captures.
2. How to search in a new higher-order representation of a solution
3. What can searching in higher-order representations do that other algorithms cannot, (i.e., assuming a single level representation)

In this section the SOTA MBOA and DO functionality are compared to provide clarity on characteristics that distinguish the performance between them. Hypothesis are created for the types of problem challenges that provide a clear differentiation between an algorithms performance. In addition, this chapter provides clarity for the contribution DO makes to the class of MBOAs.

As we discuss, there exist many design decision that can be made, however, results that demonstrate an algorithms performance, in comparison to the other algorithms, does not extend beyond an efficiency difference. Specifically, identification of problem challenges that differentiate the performance to a can solve to cannot solve distinction. Considering the large differences between the algorithms this is an unsatisfying result. For instance, a Bayesian network is capable of representing complex multivariate dependencies whilst the link is tree can only represent a simple tree structure yet performance evaluation have failed to distinguish a difference beyond efficiency. Thus, a model with higher capacity has failed to show that it can exploit problem structure that models with a lower capacity cannot.

This chapter addresses this issue using a framework of induced neighbourhood search. All state-of-the-art MBOAs can be described as algorithms that induce new neighbourhood representations of a search space and perform local variation in this neighbourhood to improve a solution. In doing so, provides a framework for us to compare functional difference between the state-of-the-art algorithms, draw hypothesis to distinguish the performance and conclude that all state-of-the-algorithms should be included when comparing the performance of MBOAs.

The SOTA methods reviewed in Chapter 2 all share the idea of exploiting relationships that contribute to the quality of a solution to enhance the search. However, their functionality, or there evolutionary connection differs: hBOA - population dynamics, MSSA - Developmental dynamics

We use this analysis to generate hypothesis for the types of problem challenges that will provide a satisfactory differentiation between the algorithms performance.

previous literature refers to p3, as being separate and not compared Hsu and Yu (2015), where as we see P3 as an example of inducing a multi-level representation. In addiation, the use of local search to initialise a EDA is considered a hybrid technique Hauschild and Pelikan (2011), where as we see this as variation and selection performing at a lower level than the single layered model. The introduction of DO provides an algorithm that can utilise these approach's, therefore providing an algorithm that connects these often separated ideas.

Model-Building Optimisation Algorithms (MBOAs) are a class of optimisation algorithms that use a machine learning to construct a model that captures the relationships between variables that contribute to a solutions quality. These relationships are then exploited to enhance the search process.

MBOAs navigate this landscape using processes inspired by natural evolution. Generally, MBOAs apply a hill-climber selection, i.e., a candidate solution is only replaced by an alternative solution with greater fitness. The method of selection could be modified from strict hill-climbing in a number of ways, such as novelty search Lehman and Stanley (2011). However, an MBOA overcomes challenges in the landscape by adapting the neighbourhood of the search space using machine learning methods. The model captures relationships between variables from a distribution of promising solutions to compress the dimensionality of the search space. This compressed representation is then exploited to improve candidate solutions.

Identify three main distinguishing factors between the alogrithms. These are:

- Model capacity - the ability to capture and represent relationships between variables at teh solutino level
- Model exploitation - the method used to adapt solutions using the capture infromation
- Model Construction

Therefore not only is the design of an algorithm dependent on representing these structure (model capacity) but in also identifying the using selection methods (model exploitation)

thus our focuses is on understanding the performance difference between the machine learning models used in the SOTA algorithms - Specifically, the capacity of the model and how the information in the model is exploited to advance future search. There exist many version of MBAOs, each with using different models and methods for utilising the models for search. The first main contribution of this thesis is to distinguish the performance between MBOAs with specifically regards to the types of problem structure they can and cannot exploit

4.5.1 Model Exploitation

The model used determines the methods available for exploitation (i.e., BOA and sub structural search isn't natural) Representation learning provides a natural way of search in the space of partial-solutions. NN can be interpreted in both cases.

to improve the average quality of a distribution of solutions

EDAs use the model to generate new solutions by sampling a probabilistic model - random recombination of regularities observed in a distribution of promising solutions - a process we term model-informed generation (MIG). MSS use the model to re-scale the variation operator to higher-orders of organisation - updating a candidate solution using higher-order variation operators - a process we term model-informed variation.

An important distinction from model-informed variation is that a new candidate solutions is not biased by individuals from the population. As is the case in two-parent crossover. Instead, for MIG the generation of a new solution is biased by the distribution of the population. A disadvantage on the other hand is the that this approach can cause difficulties with diversity maintenance.

All multi-scale search algorithms use machine learning to adapt the variation operator that is applied to a solution to adapt the neighborhood space of a solution. DO, instead, learns a representation of the solution space to adapt the neighborhood space. The variation applied at the representation is open to interpretation. This can be the same variation operator as applied at the solution level, an encoder variation operation, or crossover and mutation but at the latent representation level.

hBOA applies Restricted tournament replacement, and therefore replaces a solution that is most similar in hamming distance only if it provides a fitness improvement. This is analogous to local search, make a neighboring change to a solution and only keep if the change has provide an improvement. However, this is only try when hamming distance is a true measure of neighbouring space.

Niching niching provides a method of diversity maintenance. Therefore relative to the solution it can be observed as a type of individual improvement - more inline with MIV

Diversity of solutions that contain useful information about the problem structure Goldberg et al. (1987).

Diversity maintenance using restricted tournament replacement approximates the process of neighbourhood search redefined by the model. Therefore, BOA does not fit this framework as a solution can be replaced by any solution generated from the model. This allows for large jumps in the search space that have a larger portion of the distribution. This is not good bias as these areas of higher density may be simply easier to get to. The evidence that an algorithm that is capable of producing all solutions and restricts the likelihood during search works best by performing a neighbourhood search is interesting.

A solutions trajectory only every increases in solution quality. - from the perspective of an individual solution - we can then talk about hBOA. ——-

MIG Quality of a solution monotonically increases (from the perspective of a solution), rather than favouring average fitness

In EDA, a solution can be potentially replaced by a lower level fitness, or in a different basin of attractor. RTR limits this from happening in the population.

Estimation of distribution algorithms aim to model the structure of allelic associations in a population of individualsat a single level of biological organisation. Exploiting the problem structure to update a distribution of solutions by implicitly exploiting regularities that contribute to a solutions quality via random mixture

Model-informed generation is defined as generating a complete solution from the model by sampling the model. The sampling method is specific to the representation used by the model to capture the joint distribution of the data. Therefore, the success of MIG is determined by the model's accuracy in capturing the conditional probabilities.

hBOA uses binary tournament selection to between two random solution drawn at random from the distribution of solution.

The disadvantage of MIG is that it requires learning ‘how’ to move in the solution space. The model must capture dependencies between variables that produce correlated movement in the solution space. In the cases where this signal is weak, this can lead to difficulties in capturing these statistics and therefore require a large training set. A more significant disadvantage occurs when the signal to improve a solution is not the same signal that forms a higher-order solution

The advantage of MIG:

- Is that it applies parallel search. Only a single function evaluation is performed to decide which direction to move a solution.
- We can concentrate our search in high fitness densities: Replace solutions that are deemed not useful with variants generated in places of high densities, thus expanding our exploration in the regions of high fitness densities.
- The model representation can be black-box: The desired properties of the representation are limited as long as the sampling technique reproduces a similar distribution of solutions learnt.

The disadvantage of MIG:

- Unable to explore the neighborhood of a solution before replacement: We must make a movement in only a single direction before the model recaptures the statistical dependencies
- Requires learning how combinations go together

MIG therefore requires the model to learn how the building-blocks go together to avoid random search between them to find the combination of building-blocks that maximises the solution quality. We therefore hypothesis that random recombination will fail when signal to improve a solution is not the same signal that forms a higher-order solution

EDAs use a probabilistic machine learning model to capture the joint distribution of a training set by learning conditional dependencies between the original solution variables. The most sophisticated method uses a decision forest to represent conditional dependencies of a Bayesian network (hBOA). As the model is probabilistic, it is restricted to model-informed generation only. There is a large amount on research about the performance of probabilistic sample and replacement strategies. For a comprehensive review, the reader is directed to citeEda survey. A large topic is niching – maintaining diversity of solutions in the distribution of solutions. As discussed in the technical details for MIG in DO, DO allows for automated control. Simply it involves replacing solutions based on their fitness as well as neighbourhood distance calculated at the solution level representation (generally the hamming distance). We can interpret this as making movements in the neighbourhood space, and as such the model must learn how to do this, it cannot explore its neighbourhood, it can only generate random points in the neighbourhood space. The Algorithm:

Therefore mechanisms such as the selection pressure can be modified to control the selection pressure for prioritising promising candidate solutions during recombination. During, a genetic algorithm run, the dynamical process changes due to changes in the variation available and type of selection pressure. For instance, fitness proportionate selection enables allows for all solutions to be selected for recombination. However, when fitness differences between a solution are small, there is negligible prioritisation of solution, and therefore exploitation is halted. Truncation selection on the other-hand does not fall into such trap. It ranks the population of solutions according to their fitness and selects the top p solutions for recombination. Therefore solutions with low quality will not be used for recombination. This can be disadvantages because, some of these solutions may contain useful variable combinations. Tournament selection provide a tunable selection pressure between the two. Tournament selection selects the best solution out of a random sub-set of k-individuals. When k= the size of population, it become elitist selection. When k=1 it become random selection (thus no selection pressure). Thus in tournament selection, it is possible for all solutions to be used in recombination. Larger tournament sizes increase the selection pressure for higher-quality solutions to be exploited.

One of the more challenging limitations of genetic algorithms is diversity maintenance. Crossover operators only provide the linkage information. Specifically, which variables are interchanged between two solutions drawn from a population. Therefore, the a new variant is also dependent on the variation in the population.

RTR by performing a competition – a comparison of the fitness with - with its closest neighbour as calculated by the hamming distance. The solution added to the new generation training set is the solution with the highest fitness. At a high level of description, one can describe this as a process of trying to approximate Model-Informed variation - making partial variations in the direction of increased fitness.

Therefore tournament replacement removes selection acting on the group level and instead apply implicitly at the individual level. In MSSA, selection is explicitly applied to the individual level.

Model-Informed Variation

Search is biased, controlled by the current solution - thus reducing the issues related to diversity maintenance. Thierens and Bosman (2011)

Exploiting the problem structure to update an individual solution

Selection is used to immediately accept or remove information.

Selection provides immediate feedback to any update perform in DO. Therefore the effects of a spurious or poor representation can be mitigated by selection. Selection will reinforce relationships that have been well represented by the model, and remove/discourage spurious or poor relationships.

O

A Multi-scale search algorithm (MSSA) on the other hand allows for deterministic recombination.

It is assumed that the higher-order partial solutions are a decomposition of many low-order partial solutions. If this is not the case, and a higher-order solution is a random combination, that shares no relationship with the lower-order partial solutions, then DO it is hypothesised that DO will fail to find the global optimum solution.

4.5.2 Model Capacity

In both EDAs and MSS algorithms, the solution is represented as a single level of organisation.

The use of a deep neural network presents two interesting ideas. The first is that the model is capable of representing a hierarchical representation of the features that contribute to a solutions quality. The second is that deep layers facilitate the learning o more abstract features from data-set that are components of the lower-level features. With this insight, the DNN represents a multi-level organisation of the solution analogous to the multi-level multi-cellular organism organisation. SOTA methods do not allow for this

For EDA, model used to prioritise a robust method for sampling from the model. It is for this reason we hypothesis that neural networks to compete directly with a Bayesian model fail to to be competitive.

For MSSA, representation that captures the relationships between the directions of variation. The model bias encourages the representation.

for DO. Linkage learning is the information contained in a weight matrix. If you converted the weight matrix into a 1 and 0 matrix representing if it is or is not connected then that would provide a crossover mask.

Model capacity refers to the ability of the model to capture and represent relationships between variables.

The model used in MBOAs must be capable of separating the location of the building-blocks and representing the building-blocks efficiently.

State-of-the-art performance is achieved using non neural-network models in domains outside of optimisation.

Representation limitation of rHN-G - limited to pairwise correlations DO - Units that appear far apart in solution space are represented as neighbours in a new representation space.

Probabilistic – EDA's

Deterministic – Dependency structure matrix GA, rHN-G, DO

Graphical Model vs Tree Model

Representation of relationship - linkage only information, variable assignment

We expect differences in model capacities to have a significant impact on the adaptation that an MBOA can exhibit.

More prominant is the difference between the models used in state-of-the-art methods. Namely, performance comparisons show that less sophisticated models, such as using a linkage tree, outperform more sophisticated models like a Bayesian network with regards to the number of function evaluations performed to find a global optimum Thierens and Bosman (2013); Goldman and Punch (2015); Hsu and Yu (2015). We hypothesis that this is due to inductive bias of the model being better aligned with the problem structure. i.e., the problem structure is a tree structure. Therefore a model that has a strong inductive bias to represent a tree representation of dependencies will be better aligned than a graphical model to capture this.

Hypothesis: Tree vs Graph structure - LTGA and P3 cannot represent overlapping dependencies between variables/building-blocks.

There are two clear distinctions between the models used. The first is a tree (pairwise) vs a graphical model (multivariate). P3 and LTGA are the only algorithms that use a tree structure to represent the dependencies. The fundamental differentiation between a tree and graph model is the ability to represent dependencies when dependent variables share independent variables, as illustrated in figure 1. Alternatively, more concisely, overlapping dependencies.

This distinction was made early during the development of EDAs Bosman and Thierens (1999); Pelikan et al. (1999). However, benchmarks containing overlap used to demonstrate the performance of multivariate models have recently been shown to only cause polynomial scaling when using LTGA or P3 Goldman and Punch (2015); Hsu and Yu (2015); Chen et al. (2017). If the overlap in these benchmarks indeed created multivariate dependencies, then we would expect MBOAs using tree models to fail. We hypothesis that the dependency structure in these benchmarks, while containing overlap, are approximated sufficiently well using a pairwise model. We, therefore, revisit the definitions of overlap and develop the idea of cooperative overlap to overcome this limitation.

The second clear difference between models is nonlinear hidden variables. DO is the only MBOA to use nonlinear hidden variables. The neural network is a graphical model that uses nonlinear variables to represent dependencies between variables. Nonlinear hidden units can reduce the number of dependencies required to represent the relationships between variables by transforming the representation into a different space. A hidden unit can represent a nonlinear interaction between variables. In doing so allows for efficiently uncovering and representing the dependencies between nonlinear interactions. Representing this relationship in the original solution spaces will require an exponential increase in the number of dependencies. In machine learning, we can see this as the difference between using a neural network with linear hidden variables (PCA) and a neural network with nonlinear hidden variables (Autoencoder), or a Bayesian network with or without hidden nodes.

4.5.3 Model Construction

EDA's use pairwise dependency to construct the models. DO makes no such assumption and therefore for is more general and capable of modelling k-wise dependencies. Cost of learning and construction

Linkage Learning - EDAs probabilistic Model - MSSA/ LTGA - Deterministic model - correlation learning or a model of the linkag information - DO Induced from salient features from the dataset

Models that do not have a notion of depth, such as those used by the state-of-the-art MBOAs, do not have an alternative.

4.5.4 Multi-level representation

In both EDAs and MSS algorithms, the solution is represented as a single level of organisation.

- Why all algorithms delete the model and not incremental - new W is more powerful.

A key difference in MBOAs is that the model is regenerated after each generation. Therefore each dimensional reduction can be different from the dimensional reduction applied previously.

A distinguishing property of Deep Optimisation in comparison to all SOTA-MBOAs is the use of multiple hidden layers to represent the problem structure. DO enables us to ask the question, how does a multi-level representation of the problem structure effect the performance of search. Using the framework of DO, and multi-scale search, we can explore this using an intuitive example.

Hidden neurons = non-linear separability of variation

Allows for local search at multiple neighbourhoods, high level search can enable a solution to move into the a better basin of attraction and lower-level search can move the solution towards the optimal solution within the basin - repair

Figure 4.10 - if variation is no longer required at the lower order features, the higher-order feature is an additive combination of the lower order features. Therefore, we can simply represent this with a shallow representation - there is no requirement to maintain the information for how the higher-building block is construction. Variation is only required at the highest level representation. However, in the case of an overlapping low-order features that combine to higher-order features, variation between the lower-order components is present and therefore will require a multi-level representation.

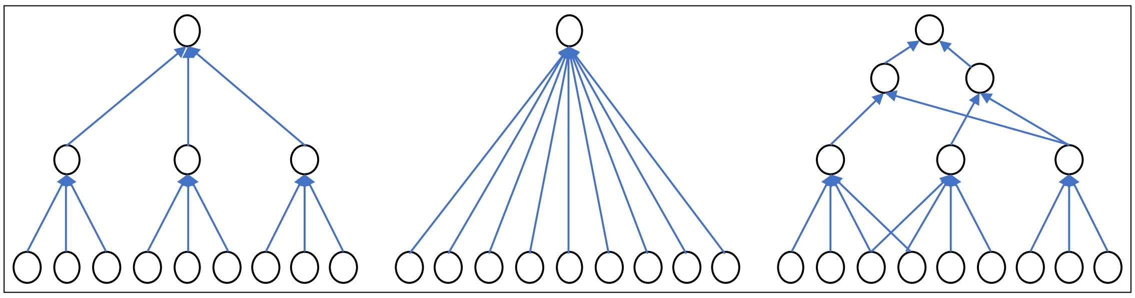


FIGURE 4.10: The need for a multi-level representation - overlap between lower-order features

Throughout the development of DO it have become clear that, in general, it is best to favour a model that under-fits the data as this maintains the degrees of freedom during variation. This is inline with other EDA as they use metric with an implicit parsimony pressure. In DO we can have more direct control of this pressure via the hyperparmeters.

4.5.5 Transfer Learning

Neural network models are amenable to transfer learning Significant research in this area - useful for BBO and especially in the real-world where problems require repeated optimising with subtle variation to the constraint values etc

Transfer learning (also related to muti-task learning) referes to the learning task of extracting and representing common features between difference (but related), enabling a sharing of knowledge between tasks and therefore reducing the effort in ‘relearning’ the same relationships.

Learning a good representation is often associated with transfer learning as if one can extract the features of the dataset, then its likely that higher-order computation with a subset of these features will acheive the learning task.

An noticeable distinction between DO and the other SOTA MBOA methods is that DO constructs and maintains the multi-level representatino of the problem structure. Lower-level structure, that was used during the optimisation process is maintained in the model. Therefore, it is hypothesised that DO can extand the capability of MBOAs to dynamic optimisation probelms, repeated optimisation where the problem structure is similar in each instance.

This hypothesis seems confirmed by a number of empirical results showing the strengths of representation learning algorithms in transfer learning scenarios.

We demonstrate the ability of DO for repeated optimisation in Chapter 7

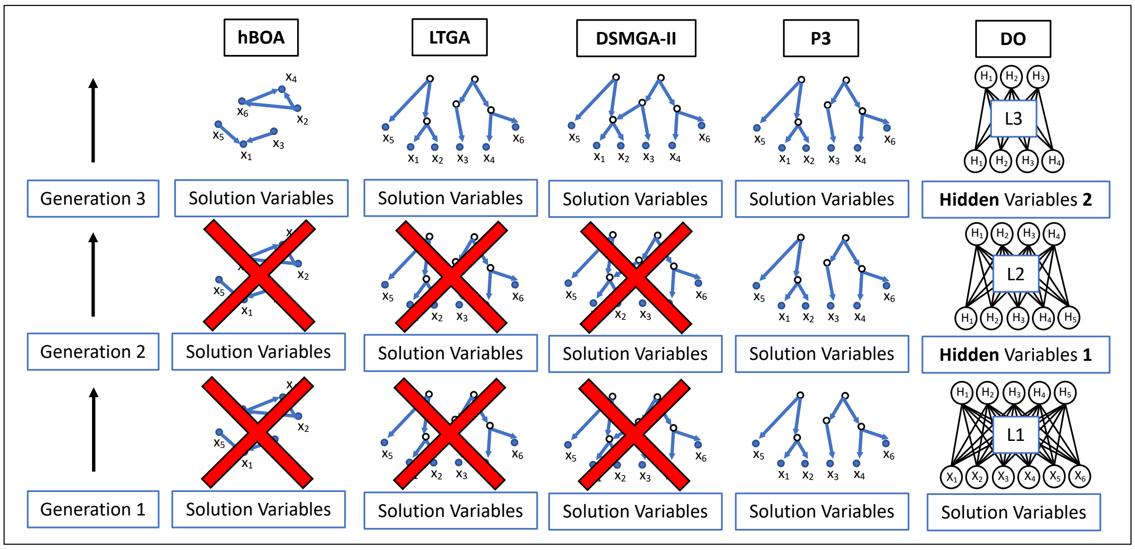


FIGURE 4.11: Comparison of the models and how the models are used by MBOAs to optimise a solution to a problem

4.6 Summary

State-of-the-art (SOTA) MBOAs Pelikan et al. (2003); Thierens and Bosman (2013), however, do not use a neural network model. The MBOAs that do are generally limited to a single layer and produce uncompetitive results Churchill et al. (2014a); Probst (2015). A summary of MBOAs using neural networks can be found in Santana (2017). DO differs from these approaches in two important ways. First, DO constructs a deep representation of the solution by recursively transforming the solution representation. Each layer is constructed using solutions that are locally optimal relative to the neighbourhood defined by the preceding layer. Second, previous MBOAs that use a neural network model generate complete solutions from the model — Model-Informed Generation (MIG). In contrast, DO improves a solution by explicitly searching in a latent representation of the solution — Model-Informed Variation (MIV).

We need not compare with autoencoder methods in the MBOA literature because they have not been shown to be state-of-the art for both shallow and deep implementations. Different architecture is interesting for future analysis.

Chapter 5

Synthetic Optimisation Problem

Exploiting the regularities in an optimisation problem class generally requires knowledge about the problem structure. In many optimisation problems, its no possible to understand the problem structure. In this case, we refer to these optimisation problems as black-box functions - we can calculate the quality of a solution, but we do not know what contributed to the solutions quality.

Of course, exploiting the natural structure may not align with the actual problem structure (the interactions between variables that contribute to the global solution), and in this case, finding the global optimum solution will take exponential time. In the case that the alignment is correct, then the global optimum solution can be found in polynomial time. Between these extreme cases, a ‘good’ quality solution can be found by exploiting natural problem structure.

In complex cases, the relationship between the solution and its quality cannot be mathematically represented. Either, the representation is infeasible, or simply the relationships are unknown (such as the case in simulations). These types of optimisation problems are called black

In some cases it is not possible to represent an optimisation problem using a algebriax modelling language. This can be due to it being intractable (the numebr of functions grows exponentially) or simply it is not posible, as in the case of simulation based problems

The development of MBOAs is driven by the identification of challenges that are present in the search space caused by the problem structure.

The appeal of MBOAs is their applicability to multiple domains and problems with unknown problem structure. With this, it is difficult to understand the performance of the algorithm to ’real-world’ (or more realistic) black-box optimisation problems because the structure of the problem is not understood or known. Therefore, to understand the performance of these algorithms, it is necessary to simulate problem challenges by explicitly controlling the problem structure within the black-box. Researches use theoretical optimisation problems to achieve this. It is via this approach that has lead to the development of MBOAs.

For global optimisation, it is therefore required that the optimisation problem has optimal sub-structure - the global solution can be constructed out of local optimal solutions.

We determine the success of an algorithm by measuring how the number of function evaluations performed to find a global optimum scales with the complexity of the problem. A successful algorithm has a polynomial relationship An^c and a failing algorithm has an exponential relationship Ac^n, where A is the coefficient, n is the size of the problem and c is constant. We refer to efficiency differentiation when algorithms have the same base and differences are only in the coefficient or constant.

The performance of an algorithm can measured in multiple ways. Generally, the most important measurement is computational effort required to find a solution of certain quality. Algorithms that have theoretical guarantees for finding the global optimum solution, such as exhaustive search, are computationally infeasible for problems beyond the most simplistic cases. On the other hand, algorithms that do not have a theoretical guarantee, such as heuristic methods, can find ‘good’ quality solutions, with significantly less computation effort compared to exhaustive search.

An algorithms performance can be measured as the ability to scale to higher-dimensional problems. For small problems, the difference in computational effort, given today’s computational power, can be insignificant and thus simplistic methods preferred. However, as the problem instances increase in dimension, the simpler methods can have poor scaling compared to the more complex methods. As such, these more complex methods can have greater performance. Scaleability also provides a measure of what an algorithm can do and cannot do. An exhaustive search scales exponentially as the number of dimensions for a problem class increases. Therefore if an algorithm scales exponentially, then it is unable to overcome the problem challenges present in the problem class. The algorithm relies on a process of exhaustively searching the solution space at some point during the search. Of course the algorithm may still outperform a simple exhaustive search with regards to efficiency (a change to the exponent value). An algorithm that scales polynomially with the number of dimension shows that the algorithm is able to overcome the problem challenges presented in the problem class. The algorithm is well aligned with the problem class. Therefore, it is said to have greater performance, even if for problems of lower dimensions the absolute difference in computation effort greater than the simpler algorithms.

Additional performance measurements include domain application and useability of an algorithm. There exist an extensive set of methods for searching in a solution space of an optimisation problem. Developing optimisation methods that align with a problem class requires intelligent design to exploit the structure of the problem. T

The only failure success distinction present in the literature is pairwise indepdnece problems as it exploits a limitation for how the multivaritte models learn (constructuted.). However, this type of problem charactersitics is often dismissed, generally due to the syntheitc problem being unique and a lack of clarity of what cause a pairwise independent function.

Benchmarks have been used throughout the development of model building optimisation algorithms. it has been important to explicitly identify problem challenges that differentiate the performance between algorithms. It is important to note the separation between problem characteristic and problem challenges. a problem characteristic is a specific reference to the structural property of a problem, for example, dependency structure, modularity, hierarchy. A problem challenge is a result of a problem characteristic or combination of problem characteristics that an optimisation algorithm must overcome. For example, the characteristic of hierarchy presents the challenge of rescaling the variation operator to higher orders of an organisation such that low order solutions are conserved during a future search.

in this paper, we employ the term sub-functions. If an optimisation problem contains possible decomposition then the simplest form is called additive decomposable functions. these functions contain sub-functions that have no interaction between sub-functions. as such optimising, each individual sub-function provides the globally optimal solution. we most difficult case is where no decomposition exists and therefore there exists no meaningful dimensionality reduction of the problem. in between, we have partially decomposable optimisation problems.

MBOAs on general optimisation problems Optimisation problems containing random or deceptive structure will always cause difficulties for MBOA. MBOAs assume that the correct problem decomposition can be identified from solutions with good fitness. A problem that satisfies this assumption is said to have optimal substructure. Given a problem that satisfies this assumption, the differentiating performance between MBOAs will be attributed to the model bias. Due to the no-free lunch theorem, models that have strong bias will be efficient at learning a restricted subset of structures compared to a model that has a weaker bias and is more general. Due to the no-free lunch theorem, MBOAs will only be successful on optimisation problems that contain optimal sub-structure.

It is important to have clear and controllable structure such that performance differneces between algortihms can be clearly attributed to specific problem challenges and problem characteristic. An important propety for any benchmark is have a calculable global optimum in polynomial time. This reduces the noise in experimentation. Further we can gain a greater understanding on algorithm performance as we can measure the distance away from the global optimum.

We can evalute the performance of an algorithm to be succesful or fail on an optimisation problem by performing an scailing analysis. For algorithm that fail, as the complexity of the problem characteristic that cuases failure increases, then the algorithm will scale exponentially with respect to the increase in complexity. It is therefore important to have a problem that when the complexity is positively proportional to increase in complexity. Further it is important that when increasing the complexity of a problem the difficulty of the problem remains relative to the size. For instance, if increaseing the complexity of the problem requried a increase in the number of gloabl optimum, then the problem may actually become easier (parity functions, as k increases the number of global optimum increases).

Hierarchy Hierarchical Problem Solving: Pelican and Goldberg discuss that algorithm design to successfully solve hierarchical problems it must address three key areas: (Pelikan Goldberg, A Hierachy Machine: Learning to optimise from nature of humans, 2003) (Pelikan Goldberg, Hierachacal Bayesian Optimisation Algorithm, 2006)

Decomposition: Must be able to correctly decompose the problem at different levels of hierarchy

Chunking: Capable of representing partial solutions at each level of hierarchy

Diversity maintenance: Able to preserve alternative partial solutions until it becomes clear which partial solution is correct. (Niching)

Niching: Discovery of multiple solutions of the problem and preservation of alternative solutions until one can decide which one is better. Without niching a population can be suspect to genetic drift which can destroy some alternative solutions which could be the ones that we are looking for. All niching strategies localise competition be solutions in some way.

The characteristics of the landscape present challenges that an algorithm must overcome to navigate to a superior solution efficiently. We are interested in methods that navigate through the fitness landscape in order to find a globally optimum solution despite these challenges.

5.1 Introduction

Theoretical problems which allow specific understanding of its performance

5.1.1 Claims

1) Development of a theoretical problem that allows for clear, independent and proportional control of problem characteristics that differentiate the performance SA-MBOAs into cannot solve and can solve categories. 2) The problem challenges that are sufficient to differentiate the performance are: a. Non-linear solution trajectory: All MBOAs cannot hill-climb using the model. rHN-G and DO can. (Model Exploitation – the way information is used by the model). b. Non-linear dependencies: Linkage learning MBOAs cannot represent non-linear dependencies, hBOA and DO can. (Partial Solution Representation) c. Pairwise Independent dependencies: All MBOAs cannot learn pairwise independent functions. DO can. (Model Construction). d. Non-orthogonal variation: All MBOAs using a tree to represent the problem structure cannot perform non-orthogonal variation. (Model Representation).

5.1.2 Abstract

The idea of formulating complex high-order organizations from a nesting composition of more straightforward organizations is a founding component to the development of Model Building Optimisation Algorithms (MBOAs). These algorithms attempt to automatically identify and model the natural decomposition's in optimisation problems. Partial solutions found for the decomposed problems are then combined to improve the solutions. If a problem has optimal structure and the MBOAs can learn and exploit the structure, then MBOAs can efficiently search the solution space to find the global optimum. Therefore, the type of model and how the information is used to search the solution space should have a profound effect on the suitability of an MBOA on a problem. If the MBOA cannot learn, represent or exploit the information then the algorithm will fail. Considering the difference between the models used in state-of-the-art MBOAs (SA-MBOAs) and differences between how the model is used to improve the search. It is unsatisfying that there exists no theoretical benchmark that provides a differentiation between the algorithm beyond efficiency. Therefore, in this chapter a theoretical problem, called Structured Variation Problem (SVP), is developed that allows for independent control of problem characteristics that define the directions of variation and the fitness landscapes. If an MBOAs can learn and represent the direction of variation and exploit this information to follow the fitness gradient, the global optimum solution is found in polynomial time and therefore successful, otherwise it will take exponential time and therefore has failed. The problem successfully differentiates the SA-MBAOs into can solve and cannot solve categories. The problem challenges identified to causes this differentiation are non-orthogonal variation to differentiate between a tree and graph model, non-linear solution representation to differentiate between linkage-learning and solution representation non-linear trajectory to differentiate between population search and hill-climbing and pairwise independence to differentiate between model construction

and model training. No state-of-the algorithm is successful on all problem challenges. DO is a novel algorithm that uses a deep neural network as the model that transforms the representation of the solution at each hidden layer to improve the search. DO is the only algorithm that is successful on all problems. SVP provides a framework for evaluating the performance of MBOAs and how the type of model and how the model is used can have a significant impact on the algorithms performance.

The motivation of this thesis is to understand how the model affects the type of search that can and cannot be applied. Specifically, what does a DNN enable a MBOA to do that other models, or shallow (single layered networks) cannot. This section explores the space of problem challenges that are hypothesized to differentiate the performance between SA-MBOAs and DO. The problem challenges identified are cooperative overlap, pairwise independent functions and non-linear search trajectories.

The question that have arisen from Chapter **??** are the following:

1. What problem structure requires a graph model representation rather than a tree model?
2. Does Model-Informed Variation outperform Model-Informed Generation?
3. What problem structure is a neural network successful on that a Bayesian network or linkage-tree is not.
4. What problem structure requires a deep neural network?

To answer these questions, a theoretical benchmark problem is required. The problem must have clear and controllable structure such that the performance of an algorithm can be attributed to specific problem characteristics. In this chapter a theoretical optimisation problem is developed to allow for exploring problem characteristics to answer these questions. Current theoretical problems have failed to provide results that show a polynomial vs exponential scaling between algorithms to show a can vs cannot (respectively) do performance. Further, the theoretical problems are separate and therefore attributing the performance of an algorithm to a specific problem characteristic is not explicit as accidently challenges induced by different problem definitions. Therefore, the developed theoretical problem overcomes this. The theoretical problem allows for independent control of problem characteristic using a single framework. Further, the problem structure is clear allowing for explicit attribution of problem characteristics that MBOAs can and cannot solve. An important challenge in developing a theoretical optimisation problem is to design a problem that scales with the complexity of the challenge of interest. Therefore, the theoretical problem is developed to that not only separates the problem but that increasing the size of the problem causes and exponential scaling to algorithm that fail to overcome the challenge. If the algorithm can overcome the challenge, finding the global optimum is easy and therefore scales polynomially. Considering the differences between these algorithms, both in terms of model used and methods for exploiting the information, it is surprising that problem characteristics that an algorithm can handle that others cannot has not been presented clearly in the results. This opens the question of, is it necessary to have a powerful model like a Bayesian network, is it necessary to use a graph representation of the problem structure. If not, then this implies that a deep neural network will not improve MBOAs. In the machine learning community, there is a clear differentiation between these types of models in terms of what they can do and what they cannot do. Generally, the more sophisticated the model, the more complex the structure they can represent at a cost of time and or data to construct the mode. Yet this differentiation does not exist in MBOAs. Therefore, there is evidence for why this differentiation should exists in MBOAS. This chapter provides this differentiate. This chapter first starts by exploring the literature to identify problem characteristics that create challenges to MBOAs. Then a theoretical problem is developed to allow for independent control of the identified problem characteristics. The performance of SA-MBOAs are the evaluated using the theoretical problem. The theoretical problem is explored to find the problem challenges that cause SA-MBOAs fail. Finally, the results are discussed to improve SA-MBOAs and how an neural network will provide a model that can overcome these challenges.

5.2 Literature Survey

5.2.1 Definitions

Neighborhood – the neighborhood of a solution refers to solutions that are accessible from the current solution. Each accessible solution is called a neighbor. Collectively they form the neighborhood. The neighborhood is dependent on the representation of the solution and the variation operator used to move in the space. The neighborhood is independent of the fitness function. **Fitness Landscape** – the fitness landscape is an abstract representation of the fitness gradients on given a solution space. The landscape is dependent on the solution space representation. **Problem structure** – The structure of the problem refers to the hidden relationships that define the path from a random solution to the global solution. **Problem Characteristics** - A problem characteristic refers to the internal structure of the dependencies. A characteristic is a specific element of the problem structure, e.g. hierarchy. **Problem Challenge** - A problem challenge is what the optimizer must be able to overcome in order to search efficiently. There can be multiple problem challenges to overcome in a single problem and a problem challenge can be a product of multiple interacting problem characteristics. For example, the problem characteristic of non-additive separable problems presents the challenge of diversity maintenance. The problem challenge of hierarchy presents the challenge of rescaling the variation operator to high orders of organization.

5.2.2 Benchmarks

Evaluating the performance of a novel algorithm generally follows the following pipeline: Firstly, performance evaluation using theoretical problems with specific problem characteristics to differentiate the performance between other state-of-the-art methods. The performance is generally measured by a scalability study, where the differentiation between an algorithm able to solve and not able to solve is defined as a polynomial scaling vs exponential scaling performance. This provides a unique theoretical understanding on the types of problem characteristics the new algorithm is specialized for. Further, provide researchers with clear understanding for further development. However, good performance evaluation requires time in understanding the fundamental differences between the algorithms and a clear theoretical problem. Secondly, performance evaluation on a test suite of domain specific benchmark problems such as MaxSat, Knapsack, Travelling Salesman problem. These problems have unknown problem structure and therefore understanding the difference between algorithm performances is challenging. Further, in the domain of MBOAs, which rely on the assumption of optimal sub-structure, the problems are often randomly generated. Therefore, the performance of MBOAs may not be a fair reflection off the algorithm's true performance. For instance, Pelikan found that hBOA performed better when the Maxsat problems where not randomly generated citepelikan2003hboamaxsat. Therefore, comparison between the solution found by other MBOAs may be more suitable marker than the global optimum (or best-known solution). Nevertheless, it provides a suitable test bed to dealing with more 'realistic' problem characteristics and thus evaluates the applicability of the algorithm. This provides a proof-of-concept level for the algorithm if the algorithm is competitive or outperforms algorithm within the same class. Performance evaluation on real-world problems. This is of greatest importance to industry and often the result that grabs greatest attention. Often, benchmarks are over-simplifications of the real-world constraints and therefore the performance evaluated using a test-suite doesn't necessarily transfer to real-world problems. New benchmarks such as the Travelling Thief Problem have been introduced to reduce provide an intermediate step between real-world problems and benchmarks. However, applying a real-world problem to an algorithm is expensive in human-hours and therefore often left to last after the algorithm has been successful on simplified versions. For evolutionary algorithms, performance has generally been measured by developing theoretical problems that contain specific problem challenges that allow for clear differentiation between the algorithm performances. Often, these problems carry little resemblance to well-studied problems used in the OR community. However, they allow for clearer evaluation differences between algorithm performance and thus develop a theoretical understanding of the methodologies. Further, evolutionary techniques are inspired by natural evolutionary process. Therefore, understanding the characteristics an evolutionary algorithm can and cannot solve provides valuable insight into understanding how the evolutionary process can affect natural evolution. In the Operations Research community, benchmark problems consist of simplified versions of real-world problem that contain constraints that are present in real-world problems. For example, infeasible solutions. Often these problems are randomly generated to produce a benchmark suite that enables comparison between algorithm performance. In these cases, the problem structure is not controlled, nor well defined. Therefore, understanding the reasons for the differentiation between performance is challenging. Further, these problems are generally NP-complete and therefore the performance is not evaluated in terms of scalability, but rather the best solution found, time to find the best-known solution, or the maximum size of an instance. The state-of-the-art algorithms are biased towards problem structure rather than the problem domain. Neither theoretical problems nor benchmark problems should be ignored. The advantages and disadvantages are clear. A theoretical problem allows for clear differentiation between algorithm performance that enables a level of understanding of what methods can and cannot do. However, in identifying problem challenges that algorithms can deal with that others cannot opens the question of how important and or abundant this problem challenge is in real-world optimization problems. In benchmark problems, they provide a more satisfying proof-of-concept of an algorithm that enables deployment to real-world problems. As generally these problems share some of the constraints that are present in real-world problems. However, the performance provides little understanding on why the algorithm works in comparison to others. The state-of-the-art algorithms are biased towards a problem domain rather than the problem structure. Theoretical problems are developed to present clear problem challenges to an optimizer. The problem is constructed such that if an algorithm can exploit the problem structure, the algorithm can easily find the global optimum. Therefore, the number of functions evaluations performed to find the global optimum will scale polynomial with respect to the size of the problem. An algorithm that fails to exploit the problem structure will fail to find the global optimum without exhaustively searching the space. Therefore, the number of functions evaluations performed to find the global optimum will scale exponentially with respect to the size of the problem. When an algorithm has polynomial scaling, the algorithm is said to be successful. When an algorithm has exponential scaling the algorithm is said to have failed. Therefore, a scalability analysis allows for clear evaluation of an algorithm's performance. The problems that explore different challenges are generally separated. Therefore, comparison between different problem challenges is problematic as the unique problem constructions may induce unidentified challenges that can causes a change to the scaling performance. Greater effort is being endorsed to make a single framework that combines the problem characteristics Weise et al. (2020). The research question for this thesis is how can deep learning models improve MBOAs. Therefore, identifying what problem challenges a deep learning model can overcome that models used in the SA-MBOAs cannot is required. To provides this understanding, a theoretical problem must be developed with clear and controllable problem structure such as to provide a clear differentiation in performance between SA-MBOAs and DO. As presented in Chapter 2, current theoretical problems fail to achieve this between SA-MBAOs (not including DO), thus the requirement for a new problem. The following differences between the SA-MBOAs methods have been identified in Chapter **??**:

1. Tree vs Graph Model dependency structure.
2. Linkage and Variable representation vs Linkage only representation
3. Model Construction Methods - pairwise statistics vs reconstruction error
4. Model Exploitation Methods - Population Search vs Local Search

For the remainder of this section, the literature is reviewed to construct a theoretical problem that contains the problem challenges that will differentiate these differences.

5.2.2.1 Problem Structure

MBOAs are designed on premise that problems can be decomposed into smaller easier to solve sub-problems. Solutions to the sub-problems can then be combined to form the global optimum solution. The model in MBOAs is used to automatically identify these sub-solutions from a distribution of solutions. Models are used to overcome the challenge of hand-designing variation operators in-order to search in the solution space. Therefore, models remove the challenge of random-linkage and the direction of variation. Building-blocks describe the components of a global solution **??**. A building-block can often be described as a sub-solution, partial-solution. However, this implies a limitation on the construct. More generally a building-block forms a function – a function that describes the relationships among a sub-section of the variables. These sub-functions are combined to describe the global function. Therefore, throughout this thesis, a sub-function is used to describe a building-block. In addition, a solution to this sub-function is called the state of the sub-function and a sub-function can have many valid states. Efficient search in decomposable problems require a separation of sub-functions, generally referred to as linkage-learning, and efficient representation of the states, generally a reduction (either probabilistically or deterministically) to a sub-set of states to a sub-function. How these sub-functions are combined contribute to the complexity of the problem and define how the optimization method must exploit the information. The simplest decomposition is a problem with additive decomposition. A problem has additive decomposition if the global functions is a sum of lower-dimensional functions. Specifically, the fitness of a sub-function state is independent of variable assignments (or sub-functions) outside the sub-function. Examples of such problems are concatenated trap Deb and Goldberg (1994), Concatenated Parity Function Coffin and Smith (2007). Additively decomposable problems where sufficient to create problems challenges for genetic algorithm when the linkage was not tight citeThierens1995AnalysisAD. Using Models to learn the linkage information has removed the problem challenges associated with random linkage. A non-additive problem is any problem where the state of a sub-function also depends on the state of one or many sub-function states. ?. The dependencies between sub-function states is weaker than the dependencies within the sub-function states in order to create the building blocks. A sub-function contains strong interactions between variables that are prioritized over weaker or sparser interactions between sub-functions. This differential in the dependency strength forms the structure of nearly decomposable. The presence of sub-functions generally creates local optimal solutions. A local optimum solution is one with no neighboring solution with a greater fitness and outside the neighborhood a better solution does exist. Therefore, in order to improve the solution, the algorithm must adapt the neighborhood of the solution space. In the theoretical problem, generally this consist of adapting the variation operator from individual variable to individual sub-function states. In additive problems, the gradient to the optimal state is often deceptive. Therefore, the states can be non-optimal, and the optimal state is not reachable using the variation operator, creating a local sub-optimal solution. In non-additive problems, the optimal state of a sub-function is not deceptive, however, the state is not necessarily optimal in the global function as this is determined by the state of other sub-functions. MBOAs must deal with two separate challenges. The first is learn and representing the problem structure such that the neighborhood of a solution is redefined.. This refers to accurate separation of the sub-functions and accurate representation of the dependencies between the variables in sub-functions such that the states can be generated. In linkage-learning only models, the models only learn to separate the sub-functions. - the model does not decide the value to assign to a variable. The second is exploiting this information from the model to move in the redefined neighborhood to follow the gradient in the fitness landscape. A model may be suited to representing the structure but fail to exploit the information efficiently and vice versa. Therefore, both the separation and representation of sub-functions and the fitness landscape are important challenges that can differentiate the performance. As discussed in Chapter 2, both the models and how the models are used differ between MBOAs. In order to adapt the neighborhood, the model must be capable of separating the sub-functions and representing a compressed representation of the states. In representation learning models, the compression is such that random sampling produces the state to sub-functions with high probability. In linkage learning models, the compression occurs in the population where states that do not satisfy sub-functions are filtered out. As MBOAs use different methods and models to perform this compression. It is hypothesized that the complexity of an accurate compression can affect the performance of a MBOA. In machine learning models this differentiation is made clear. There exist many machine learning methods that can be used to compress the dimensionality of a space.

The complexity of the compression or representation determines the suitability of the model. There are clear examples that differentiate the performance of the compression methods. A popular example is the difference between PCA and an Autoencoder. PCA is limited to orthogonal directions of variation in the dataset (relative to the original dimensions) where as an Autoencoder can compress in non-orthogonal directions due to the nonlinear hidden units. An Autoencoder using linear units performs a similar compression to PCA. It is therefore hypothesized that the orthogonality of the compression is one example that can have a significant impact on an MBOAs performance. Modular landscapes have a simple gradient to follow at the representation level of sub-functions. Therefore, modular landscapes require an optimizer to rescale the variation operator to sub-functions. When the MBOA has successfully done so, the global solution is easily found by searching in combinations of sub-functions solutions. Modular landscapes therefore provide a good theoretical landscape to evaluate the performance of algorithms to learn, represent and exploit the dependencies in a sub-function. Hierarchical decomposable functions define a multilevel representation of sub-functions with high-level functions constructed from a nesting of low-level sub-functions Watson et al. (1998); Pelikan and Goldberg (2001). The problems therefore have optimal sub-structure. The structure is inspired by natural bottom-up mechanisms where a problem is constructed from multiple scales of sub-functions that combine to create high-order sub-functions. The output of low-level sub functions are used as inputs to higher-level sub-functions. A sub-function only produces an output when the its constraints of the sub-function are satisfied. The solutions to low-level sub-functions provide information for the solution to high-level sub-functions. Further, finding solutions to high order sub-functions can only be achieve when solutions are found to low order sub-functions. A variation that invalidates a low-level sub-function means that all higher-level sub-functions are invalidated and therefore dependencies with these high-level sub-functions cannot be observed. Therefore, an optimizer must identify and conserve low-order solutions at multiple scale in order to search at the high order space. Specifically, high level sub-functions are decomposable into low-level sub-functions which in turn are also decomposable into lower level sub-functions. This multi-level nesting creates multiple problem challenges for an optimizer The generalised hierarchical construction method used by HIFF and subsequently HEP is detailed in Watson et al. (1998) and summarised here. A string of variables $V = \{V_1, , V_k\}$ over an alphabet S represents the solution string at the 'ground' level. p represents the number of levels in the hierarchy and N_p represents the number of building-blocks of length L_p at each hierarchic level. Each block at the first level ($p = 0$) containing k variables is converted into a low-dimensional representation of length L_p/R by a transformation function, where R is the ratio of reduced dimensionality creating a new string $V^p = \{V_1^p, , V_{N_pR/N_p}^p\}$.

Pelikan Pelikan et al. (2003) describes the following problem challenges a MBOA must overcome in order to efficiently exploit hierarchical problem structure.

1. Decomposition
2. Chunking
3. Niching

Hierarchy has been a dominate characteristics for developing MBOAs. However, these hierarchical theoretical problems do not require a deep representation to search the space efficiently - the structure can be well approximated using a shallow representation (shown in section x and Chapter 5). Therefore, the problem challenge of a multi-level representation is not present in the theoretical benchmarks in the literature. Specifically, the multi-level representation can be compressed into a single level. This is because nesting is a strict tree and the search requires traversing this tree bottom-up (there is no need to search at lower-level scales once the algorithm is capable of search at the high-level scale). Therefore, when sub-function states are correctly combined, the lower level decomposition can be disregarded for future search – it is not necessary to maintain the information of how low level variation operators combined to create a high-order variation operator to find the global optimum. Therefore, to differentiate DO from SA-MBOAs, and a deep model from a shallow (single-layered model) it is necessary for problem require the model to represent a multi-level representation of the structure in order to find the global optimum efficiently. This can be achieved in two ways. First, is to introduce problem structure that creates a nesting compression that cannot simply combine two low-level sub-functions into one – a low-level sub-function is added to another to create a high-order variation. Overlap is a problem characteristic that can provide this requirement to maintain the separation of low-level sub-functions. In overlapping sub-functions, a strict combination of one sub-function with another would not allow other sub-functions to use the lower-level sub-function to construct the other high-order variation operator. Whereas if the low-level variation operators are maintained, then constructing high-order variation operators that share low-lever variation operators is achievable.

5.2.2.2 Controlling Problem Structure

In order to evaluate the performance of an algorithm, the complexity of the structure must be controllable. The methods for controlling the complexity are either to increase the degree of complexity within a fixed sized problem Kauffman et al. (1993); Deb and Goldberg (1994); Chen et al. (2012); Chang et al. (2011); Wang et al. (2013) or to increase the amount of complexity, by have building-blocks that contain the desired complexity and increase the size of the problem to introduce more of the same Thierens (1995); **?**); Yu et al. (2005); Coffin and Smith (2007). Increasing the degree of complexity first requires a clear measure for the degree of complexity. Determining if an algorithm can and cannot overcome this complexity requires measuring the change in function evaluations to find the global optimum for a fixed size problem. As the degree of complexity increases a model that can overcome the challenges will show little or no sensitivity to changes in the degree of complexity. An algorithm that fails to overcome the challenge will be sensitive to the degree of complexity. However a measure of sensitivity does not provide a succeed and fail differentiation. A MBOA can be sensitive but still scale polynomial as the problem size increase. Further the degree of complexity can be dependent on the problem size, so as the size of the problem increases, the sensitivity to the degree of complexity can vary. Therefore, a problem that controls the problem structure by changing the degree of the problem characteristic, doesn't provide a clear differentiation between an algorithm that fails and an algorithm that is successful. Increasing the amount of complexity is generally used to provide a categorization of successful and fail off an MBOA. If the problem challenge of interest can be captured by a sub-function, then for an algorithm that cannot overcome the problem challenge and requires an exhaustive search to find the optimal solution, adding more of the same will create an exponential growth in the function evaluations to find the solution. An algorithm that can overcome the challenge will show a polynomial scaling due to the increase in problem size. A theoretical problem that provides this distinction is much more favorable. An algorithm that has exponential scaling shows that it is relying on some exhaustive search procedure to find the global optimum, it simply cannot overcome the problem challenge. Therefore, the problem challenge that an algorithm cannot overcome can be explicitly attributed to the problem challenge presented by the theoretical problem.

5.2.2.3 Overlap

Overlap occurs when a solution variable is a member of multiple sub functions that are independent at their respective level. This is a type of multivariate dependency between variables. Overlap remains a relatively under explored problem characteristic. As discussed, problem have been developed to evaluate the performance of algorithms with overlapping structure, however have failed to provide a clear differentiation between algorithm that should and should not be able to overcome the challenges associate with overlap. As previously highlighted, the idea of overlapping structure should be sufficient to differentiate the performance of linkage-tree models and all other SA-MBOAs and DO. This section reviews the research conducted on overlapping problem structure. There is some overlap with the idea of overlap and partial-decomposable problems. The definition of partially decomposable problems it that their exist different strength or number of dependencies that construct a complete system. Sub-functions contain stronger or more dependencies within than the dependencies with other variables or sub-functions. The weak or fewer dependencies between sub-functions create information of how sub-functions combine. A variable is therefore a member of multiple functions, however, its influence in sub-functions may vary - stronger or weaker. When decomposed according to the strength of dependencies the decomposition is non-overlapping if a variable is not shared between decomposed sub-functions. The decomposition is overlapping otherwise. Therefore in problems with overlap the dependency strengths of sub-functions with shared variables are the same (or similar relative to the between sub-function dependencies). The problem challenges an optimizer must overcome in the presence of overlap are **?**:

1. Separate sub-functions that allows variation between solutions to individual sub-functions to allow for combining partial solutions.
2. When performing a variation to a sub-function solution, not to disrupt other sub-functions solutions that share the same variables. Concretely, conserving the state of other overlapped sub-functions while performing a variation to the selected sub-function.

Experiments involving overlapping problem structure Pelikan et al. (2003); Goldman and Punch (2015); Hsu and Yu (2015) have failed to differentiate the performance between MBOAs using a graph or tree model representation of the problem structure. Research shows that while LTGA is less efficient than hBOA Pelikan (2010); Pelikan et al. (2011) and DSMGA-II Hsu and Yu (2015), however LTGA does not fail. Considering the complexity one associates with overlapping dependencies, this results is unsatisfying. If the overlap in these benchmarks indeed created overlapping directions of variation, then one expects MBOAs using tree models to fail. The main hypothesis for why LTGA overcomes overlap is due to the method used to construct a new solution during recombination Thierens and Bosman (2011). Or as detailed in Chapter 2, Model-informed variation (MIV). Because the model can represent multiple dependency sets and during MIV all sets are considered, the algorithm is able to incremental construct partial solutions to overlapping sub-functions. This hypothesis is further supported by the empirical evidence present in Mills thesis Mills (2010). Observed is the model further decomposes the overlapping sub-functions into variables that do and do not overlap. The MIV then assembles the partial solutions to the sub-functions using this additional decomposition. Therefore, this leads to the conclusion that the available benchmarks used to evaluate the performance of overlapping problem structure are not accurately presenting the problem challenges to the optimizer. The challenge of overlap can be overcome by an approximate method. While less efficient than accurately modelling the overlapping dependencies, as observed in the performance comparisons between MBOAs Hsu and Yu (2015); Chen et al. (2017), it doesn't cause failure. The NK landscape problem has been the primary benchmark used to evaluate the performance of overlap. The NK landscape problem Kauffman et al. (1993) is an additively decomposable problem. The objective is to maximize the function detail in Equation.1. A problem of size n is decomposed into n sub-functions. Each solution unit has its own sub-functions that also takes k additional arguments. (if $k = 0$, only the unit I is an argument of a sub-function). The sub-function transforms the arguments into a real-value, generally using a lookup table that arbitrarily assigns real values to all possible binary combinations $(2k+1)$. All sub-function outputs are summed to calculate the solution fitness. With $k >= 1$, overlap is introduced between sub-functions (A solution unit is used as argument to multiple sub-functions). Understanding the problem challenges and optimizer must overcome to solve the NK landscape is not clear. This is because, due to the random assignments of real values to combinations in sub-function, the overlap is not well-defined nor is it clear whether the overlapping dependencies are creating an overlapping structure or a non-additive decomposable function. Equation 5.2.2.3 – NK landscape

$\mathrm{F}_{nk}(X_0, X_1, , X_{n-1} = \sum_{i=0}^{\lfloor \frac{n-1}{step} \rfloor} f_i(X_{i\times step}, \prod(X_i))$

The Ising spin-glass problem is example benchmark problem to evaluate the performance of an algorithm to overcome overlapping structure Goldman and Punch (2015); Pelikan and Goldberg (2003a); Yu et al. (2009). The Ising spin-glass problem Sherrington and Kirkpatrick (1975) is defined as a set of N spins (variables which take assignments 1 or -1) and pairwise interactions between neighboring spins. The interaction defines the correlation sign a pair of spins should have. In the 2D cases, a spin has 4 neighbors arranged in directions north, east, west and south of the spin. The objective is to find an assignment of spins that minimizes the energy function, as detailed in equation 2. A single pairwise dependency can be defined as a sub-function. As such, a variable is a member of many sub-functions (4 in the 2D cases) and therefore has overlapping structure. However, as in the NK landscape problem the structure resemble a near-decomposable structure rather than overlapping sub-functions. Equation 5.1 – Ising spin glass.

$$H(S) = -\sum_{i=0}^{l-1}\sum_{j=0}^{l-1} J_{ij} s_i s_j. \tag{5.1}$$

Both the NK landscape and Ising Spin-Glass problem are unsatisfying benchmarks to evaluate the performance of algorithm when dealing with overlap. Both problems fail to ensure challenge 2. In the case of the NK landscape, the structure of the overlap is not clear therefore problem complexity cannot be specifically attributed to the overlapping structure. Finally, both problems do not proportionally increase the complexity of overlap as the problem size increases. Theoretical benchmarks have been constructed in an attempt to improve these limitations. The cyclic trap function Yu et al. (2005) was one of the first. Building-blocks containing the deceptive trap function would share two variables with two different building-blocks. The arrangement would construct a circular arrangement of building blocks, each building block overlapping with a single variable with the building-block preceding it on the solution representation. As in the normal trap-function, the problem was additive. Therefore, a solution to a building-block also has the correct variable assignment required for the other building-blocks it is overlapped with, undermining the complexity of the overlapping structure. This lead to a characterization of overlap **?**:

1. Overlapping agreement: If the optimal partial solutions for two overlapping sub-functions have the same variable states.

2. Conflict Overlap: If the optimal partial solutions for two overlapping sub-functions require different states for the shared variables Chang et al. (2011); Wang et al. (2013).

For the global solution, either an optimal solution to one sub-function is preferred causing a sub-optimal solution to another partial solution, or both sub-functions have sub-optimal solutions. With this classification, theoretical benchmarks were developed to provide a means of proportional controlling the complexity of overlap **?**Chen et al. (2012); Wang et al. (2013). The control involves increasing the size of the overlap and the degree of conflict. The constructions indeed increased the difficulty for finding the global optimum Wang et al. (2013), however the control also increased the time for algorithms that are capable of modeling overlapping interactions. Therefore, opening the question, is the increase in time to find the global optimum due to the overlapping structure? As such, complicating any interoperation from the scalability analysis. In addition, due to the under defined interactions in these problems, evaluating what the model has learnt provides very little use to understanding the performance of the algorithm. With clear structure, one can compare what the model has learnt and how it is exploiting this information with the problem structure and then better align the algorithms methodology to improve the performance.

To summarise, overlap is an important problem characteristic that presents interesting problem challenges to MBOAs. A common hypothesis is that a linkage-tree model will fail to solve problems that contain overlapping structure Thierens and Bosman (2011); Hsu and Yu (2015). Whereas models capable of representing a graph structures, such as DSMGA, hBOA and DO will succeed. Experiments thus far have failed to provide a satisfactory differentiation. An additively separable function does guarantee the requirement for preserving solutions already found for other sub functions. As long as the variation introduces more valid sub functions into a single candidate solution, then the variation will be favored. Therefore, disrupting solutions to overlapping functions can be accepted. Due to this, models not able to represent overlap take advantage of this to overcome the challenges presented by overlap. Although this may be less efficient than accurately exploiting the problem structure, it is not sufficient to cause an exponential failure to the algorithm. Further, problems containing overlapping dependencies are not clear different to a problem that is partially decomposable. I hypothesis that the primary cause for this result is due to lack of clarity between partially separable decomposition and overlapping dependencies. The overlap must refer to the variation operator, the direction of variation. Variation operators that have overlapping linkage, For instance, overlapping variation operator will occur when the required variation operators apply x1x2 and x1x3 but it is not sufficient to perform x1x2 and x3. In this case, an overlapping representation will be required. The non-additively decomposability must instead refer to how these variation operators are used to improve the quality of a solution, given the state of other sub-functions, x1x2 is the correct variation to perform, given a different state of the other sub-function x1x3 variation is required. If these can be clearly separated such that: a problem can have overlap dependencies independent of whether the problem is partially decomposable or not; and, the problem can have partially separable decomposition independent whether there are overlapping dependencies or not will provide clarity on this ambiguity. This chapter focuses on developing such a framework that allows for this separation of problem characteristics to make it clear for research's to evaluate and understand the performance of MBOAs. Further, problems have developed methods for controlling the degree of overlap and the degree of conflict. However, how changing these characteristic affects the problem challenges is difficult to measure. It is important that the problem challenge(s) presented to an optimizer being investigated by a theoretical problem can be quantified such that differentiation between algorithm can be attributed to specific problem characteristics. The review has concluded that the following problem characteristics will provide a novel and desired benchmark for the community regarding investigation of overlapping structure.

1. Proportional control of overlap complexity such that as degree of overlap complexity increases there is a proportional increase in the number of function evaluations. Exponentially proportional for algorithms that cannot solve and polynomial proportional for problems that can.

2. Non-additive decomposable theoretical problem that require overlapping variation operators – a variation to a sub-function must conserve the state of other sub-functions in order to improve a solutions quality - guaranteeing challenges 1 and 2 must be overcome to efficiently find the global optimum

3. Clear dependency structure of overlap to allow evaluation of a models performance with learning and exploiting the structure a means for measuring the alignment with the problem

5.2.2.4 Pairwise Independent

A limitation of MBOAs is the approximations used to construct the multivariate models. Namely that they are reliant on pairwise statistics to construct the model at some stage of the model construction. This is necessary as it is simply not computationally tractable to compute all possible multi-variate statistics. DO and more generally neural network models do not use pairwise statistics, the models are constructed by changing the parameters in the direction that reduces the reconstruction error. Therefore, in the presence of pairwise independent statistics, it is hypothesized that DO will be capable of learning the problem structure. Pairwise independent functions are a class of functions that have dependencies that cannot be identified using pairwise statistics. These functions are called pairwise independent functions. Pairwise independent functions represent a class of functions that contain strong dependencies only when observing more than two variables. Variables appear independent when only observing a pair. These types of functions have been identified to cause failure to all SA-MBOAs Iclanzan (2011); Chen and Yu (2009); Coffin and Smith (2007). Neural network models do not share this same limitation. It is therefore hypothesized that an neural network model will not fail. An example of a pairwise independent function is parity. Determining if the function is even or odd requires observing all variables. Observing all variable pair interactions does not provide the function value. The concatenated parity function (CPF) Coffin and Smith (2007) was introduced as a simple example to exemplify this difficulty for EDAs. It is an additively decomposable problem composed of a concatenation of parity functions. The function to optimize is as follows: $CPF(S) = \sum_(m = 0)^M parity(S_m)$ where S is the solution representation, M is the number of parity functions, and S_m is the subset of S that parity function m uses as its variable inputs. The parity function is defined as: $parity(V) = \begin{cases} C_{even} & if sum(V) is even \\ C_{odd} & otherwise \end{cases}$ where V are the variables of subset S_m. Sum is bit count of the variables and C_{even} and C_{odd} are constants. The global solution is a solution with all sub-functions even (or odd depending on the constants used in the problem). Coffin and Smith Coffin and Smith (2007) show that the CPF causes hBOA to fail. However, this problem is not difficult to solve Chen and Yu (2009). A local-search algorithm using single-bit mutation can successfully find the global solution as each module is only every a single-bit mutation away from the optimum. Chen Chen and Yu (2009) showed that cGA scaled polynomial on the CPF problem. Never-the-less the problem provides sufficient evidence that MBOAs cannot learn or exploit the problem structure or pairwise independent functions. In the cases of a non-additively decomposable problem, where the optimal state of a sub-function is dependent on the state of other sub-functions will ensure that variation must be rescaled to high-orders or organization. Therefore individual changes will not be sufficient to find the global optimum. The Hierarchical Pairwise Allelic Independent Function (HPAIF) problem Iclanzan (2011) is a hierarchical problem that uses pairwise independent sub-functions at the lowest level. Here, the authors use a hierarchical structure as detail in section [hierarchy] that combines the states of sub-functions. However, the results show ECGA is successful for solving this problem. The hypothesis for this is due to representing solutions using a binary unit. Therefore, only two solutions are required to satisfy the higher-order sub-functions, which can be accurate measured using pairwise statistics. The representation of a sub-functions state is not capturing the relationships of the sub-function. Therefore, the complexity of the sub-function is by passed and a pairwise variation is sufficient to solve the high-level dependencies. It is therefore hypothesized that the pairwise independence at the lowest level only acts as noise. Pairwise independence appears a contrived and extreme case of dependencies that is only useful for evaluating the limitations of MBOAs. However, Martins Martins and Delbem (2016) proposes an alternative explanation, other than the relationships between variables, for the occurrence of pairwise independent relationships. Martins hypothesis that a multi-model function can create the challenge of pairwise independent relationships Martins and Delbem (2016) when observing a distributing of optimal solutions. A selected distribution can appear to have a uniform dependencies caused by local optimum making pairwise statistical dependencies harder to detect. It is not that sub-functions are explicitly pairwise independent, as in the CPF and HPAIF problems. Rather, when observing a distribution of solutions, there interactions appear pairwise independent. This identification is important. The function of a model in MBOAs is to provide a method for making intelligent variations to solutions. While hBOA uses a generative algorithm, the restricted tournament replacement replicates the process of neighborhood variation. All other SA-MBOA explicitly attempt to learn the variation operator. Thus, complexity arises when the variation required to follow the fitness gradient is pairwise independent – given a distribution of solutions to a sub-function, the set of variation operator that maneuver between states is pairwise independent. Note in the CPF, the variation between each state is a single mutation and in HPAIF a pairwise variation at representation levels greater than 1. In the case of variations operators that are pairwise independent an MBOA using pairwise statistics will fail to learn the variation operator. It is hypothesized that using a neural network model will be capable of learning relationships of pairwise independent functions. The objective function used to train the neural network is not a pairwise statistic. In addition, a classical example for the power of what a multi-layer neural network can do that a shallow neural network model cannot is learning the parity function . Pairwise independent statistics is exploited in the theoretical problem to differentiate the performance between MBOAs. This problem characteristic is expected to differentiated DO from all other SA-MBOAs. Further, the sub-function itself does not need to by parity, rather, as identified by Martins Martins and Delbem (2016) the variation between the distribution of sub-functions is pairwise independent. This is only achievable when more than 2 states need to be varied between in a single sub-function. Thus a sub-function has multiple output units, or representation units, that create dependencies between other sub-functions.

5.2.2.5 Local vs Population Search

The method for exploiting information from the model is categorized into Model-Informed Generation (MIG) and Model-Informed variation (MIV) in chapter 2. MIG requires sampling the model to generate a random sample. This can induce complexities with diversity maintenance, thus restricted tournament replacement is a popular method to overcome this. RTR provides an approximate to neighborhood search, as a solution that is local in terms or hamming distance is used as competition. Effectively performing a search in a neighborhood defined by the model. MIV methods used by LTGA, P3 and DSMGA-II perform this explicitly. However, they do not learn the assignment to variables. They therefor require maintaining a population of solutions so that recombination can apply adequate assignment values to sub-functions can be made. Similar to MIG approaches, this requires a diversity maintenance in the population adequate variation. rHN-G and DO on the other hand perform MIV and model the assignment values to variables. They therefore do not require a population of solutions to perform MIV. The model is capable of determining which values to assign to each variable. It is therefore important differentiate these methods of MIG, MIV using linkage information only and MIV that represents the linkage and variable assignments. Probabilistic search has been the founding method in Estimation of Distribution algorithms. However, the additional methods, such as RTR approximate a variation method that moves solutions according to the neighbourhood defined by the model. Random generation and replacement do not achieve state-of-the-art results **?**. Further, the use of local search methods to Initialise a population has been shown, for all population based MBOAs, to significantly improve the performance of MBOAs. It therefore indicates that searching in the neighborhood of a solution is can be significant. In the case of population based methods, the assignment to a variable is dependent on the frequency of that variable in the distribution of solutions. Therefore if the correct assignment to the variable is rare or not present in the population, the search becomes trapped and will converge to sub-optimal solutions. In rHN-G and DO, a population is not required to perform variation to a solution. It is therefore hypothesized that solution trajectories that can saturate a population prematurely will causes SA-MBOAs to fail, while rHN-G and DO will be successful. For example, if the trajectory for binary variables in a solution must go back and forth relatively slowly compared to the trajectory, then it is likely the population will saturate a variable with a single value. This is hypothesis is connected to the longest path problem Horn et al. (1994). The longest path problem defines a path for local search problem that scales exponential with the problem size. Therefore a local search algorithm scales exponentially. This is achieved by artificially constructing a path that has a length exponentially proportional to the problem size is a variation operator of a single-unit is used. Because the whole search space is not used to in the path, for large instances, the path becomes aligned with a needle in a haystack problem. This is resolved by applying a slope function based on the hamming distance between the solution and the start of the long path. Therefore, solutions not part of the path are directed towards to the start of the long part. Therefore the global optimum can be found from any solution initialization using a local-hill climber. Due to the exponentially long path, a local search scales exponentially. Interestingly, a GA using crossover did not Horn et al. (1994). Thus contradicting out hypothesis that searching in the neighborhood of the solution is advantageous when compared to random recombination. However, a follow-up paper Höhn and Reeves (1996) by Horn showed that the GA was finding the solution in unexpected ways. The first method was the GA taking advantage of a royal-road structure induced in the path by the recursive construction. Following this structure leads to the global solution. The GA was therefore able to make 'short-cuts' in the path reducing the long-path to a short path. Of more interest, and the more prominent reason for how the GA found the global optimum was the insight made into the slope function. The slope functions was used to lead a hill-climber to the start of the path at 00. The slope function resembles the one-max problem. In the one-max problem, the population converges towards solutions containing all 1's as the variable assignment such that recombination between solutions can find the global optimum. As such if a GA was to follow this gradient, then it would have lost its diversity to follow the path. The GA was therefore not following the path, instead, the slope function optimum was a distance of two away from the long-path global optimum. As such the GA was generally accidental finding the global optimum of the long-path while trying to solve the slope function. While this problem still differentiates what a GA can do that a hill-climber cannot its it rather by accident than by design. Further, it is expected that an MBOA that can hill-climb using the model will be able to also take advantage of the royal-road structure and therefore, like the GA, be able to take short-cuts in the path to find the global optimum in polynomial time. The identification that a population following a slope function similar to one-max causes significant difficulties to a population based method if then a path must be followed is inspiring. If the path doesn't scale exponentially, then a hill-climber, that does not suffer with diversity maintenance, will be successful compared to a population based method. Further, if the variation between solutions along the part require a variation learnt by the model. Then an MBOA capable of hill-climbing in the neighborhood defined by the model will be successful. Otherwise, if it is unable to search in the neighborhood, it will fail due to diversity maintenance.

5.2.2.6 Incremental vs New Model Update

The final differentiation between SA-MBOAs and DO is retention of variation from a random solution to the global optimal solution. DO and P3 have additional computational cost associated with their models. hBOA, DSMGA and LTGA at each generation dump there current model and construct a new branh new model. This intuitively feels wasteful. A lot of computational time goes into constructing the model. P3 is unique in that after G generations, it has a multi-level representation of different neighborhoods. It can retain sub-structure that takes a random solution to the global optimum. All other SA-MBOAs cannot, the finally model will only contain information that takes a solution from generation g-1 to g. Is this a problem. In static optimizations, there is no evidence for why one would need to retain the information in how the search got from A to B. If we have the global solution, why does it matter if it is possible to search at higher dimensional neighborhoods. Even taking a random solution, the model can just replace it with the global solution. It becomes important for dynamic or repeated optimisation problems. When the structure to the optimisation problem is constant, but constraints redefine the global solution within that structure, retaining information of how to search at all the neighborhoods that lead to the unconstrained global optimum is beneficial. An MBOA that disregards the old information from previous generations either becomes trapped with a model that cannot find the global optimum, or it must restart the search again. This property is not explored in this section. Dynamic and repeated optimisation problem share connections with multi-level problems which are discussed and evaluated in Chapter 6.

5.2.3 Summary

Identified in the literature is areas of research that identify the types of problem challenges and problem characteristics that can causes significant challenges to MBOAs. However, the results are unsatisfying and do not clearly differentiate the performance between MBOAs. While hypothesis exists for what SA-MBOAs can and cannot do, a theoretical benchmark problems that allows researchers to explore these hypothesis does not exists. This thesis

There exists not theoretical problem that allows a combination of problem characteristics and challenges. Problems are separated, yet interactions between characteristics can create interesting problem challenges that are not exploitable using existing theoretical problems Their exists no benchmark optimisation problems that contain hierarchy and overlap.

The problem is idealised - the partial solutions are easy to identify from a distribution of solutions - a local optimal solution will contain a 100% accurate signal for the first level partial solutions. In fact, at all levels of representation - the relative local optimum contains a perfect signal.

Thesis Contributions:

- Definition of co-operative overlap - introduced by multivariate dependencies and having greater than 2 solutions to a BB.
- Complexity by non-separable functions and increase in the number of overlapping sub-functions and not the degree of overlap nor the degree of conflicting
- create a problem that must conserve alternative overlapping solutions.
- Variation must separate partial solutions and cause variation that does not disrupt the sub-function that shares overlapping variables.

- BB of overlapping functions (therefore p3 and LTGA do well because it can identify the separable BB's but cannot deal with the overlapping sub-functions that are contained within in a Building-Block)
- hypothesis that this should distinguish the performance between tree and graph based learning models.

The problem challenges of overlap and pairwise independent functions have been previously hypothesised to differentiate the performance of what MBOAs can and cannot do. However, problems containing overlap fail to distinguish the performance between graph and tree data structures. For the case of pairwise independent functions, whilst a synthetic problem exist in the literature that demonstrate that MBOAs using pairwise statistics to construct the model will fail, the problem is easy for a single-point mutation operator to solve, and regarded as a special case. An important challenge in developing a theoretical optimisation problem is to design a problem that scales with the complexity of the challenge of interest. Therefore, the theoretical problem is developed to that not only separates the problem but that increasing the size of the problem causes and exponential scaling to algorithm that fail to overcome the challenge. If the algorithm can overcome the challenge, finding the global optimum is easy and therefore scales polynomially.

Considering the differences between these algorithms, both in terms of model used and methods for exploiting the information, it is surprising that problem characteristics that an algorithm can handle that others cannot has not been presented clearly in the results. This opens the question of, is it necessary to have a powerful model like a Bayesian network, is it necessary to use a graph representation of the problem structure. If not, then this implies that a deep neural network will not improve MBOAs. In the machine learning community, there is a clear differentiation between these types of models in terms of what they can do and what they cannot do. Generally, the more sophisticated the model, the more complex the structure they can represent at a cost of time and or data to construct the mode. Yet this differentiation does not exist in MBOAs. Therefore, there is evidence for why this differentiation should exists in MBOAs. This chapter provides this differentiate.

5.3 Synthetic Optimisation Problem

Whats needs to allow for a useful theoretical problem - Known global optimum that can be efficiently calculated - Clear problem structure to enable understanding of explain why a model is successful or why it fails. - Independent control of the problem complexity that can be proportionally controlled for scalibilty analysis. - Same framework so that variations between different types of complexity can be clearly evaluated.

5.3.1 Problem Structure

The literature reviews have highlighted two key difference between SA-MBOAs. That is rescaling the variation operator (learn and representing the variation) and applying the variation operator (exploiting the information to improve a solution). To investigate these methods independently from each other requires a theoretical problem that can separate the type of variation required to move between neighbors and the fitness landscape assigned to the neighbors. This separation is often defined using a genotype-phenotype-environment framework. This framework is illustrated in Figure 1. Genotype refers to solution representation. This is the interface between the problem and the optimizer. An optimizer only apply variation to this representation. A genotype-phenotype (GP) map, called a complexity function in this problem, transforms the solutions representation into a representation suitable for the fitness function, called a phenotype representation. A phenotype-environment map, called the environment function in this problem, then computes a fitness value for the phenotype representation. The complexity function (GP map) phenotype representation and Environment function (PE map) are all hidden from the optimizer – it is a black-box function. The optimizer only has access to the fitness value. The fitness function is therefore: Eq 1 F=E(C(S) Where F is the fitness of solution S, C() is the complexity function and E() is the environment function. From this construction, it is clear that non-linearity is induced when either E() or C() is non-linear.

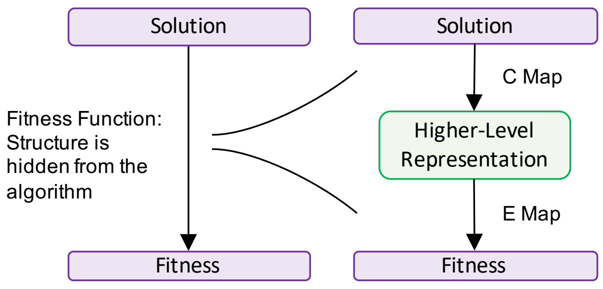

Solution
Solution
C Map
Fitness Function: Structure is hidden from the algorithm
Higher-Level Representation
E Map
Fitness
Fitness

FIGURE 5.1: We separate a solution to fitness mapping using a higher-level representation R. The C map controls the type of neighbourhood compression required. The E map controls the trajectory to find high-order components.

The intermediate phenotype representation provides a method for clearly separating the type of variation an optimizer must perform and the path to follow to find the global optimum. The type of variation an optimizer must perform is controlled by the GP map. The GP map defines how a change to the solution representation transforms to a change in the solution representation. Therefore, changing the dependency structure in this function independently controls the complexity of the variation operator from the fitness function. It is therefore referred to as the complexity function. The PE map calculates the fitness of a phenotype representation. The function is therefore defined as the environment function (as the fitness function encompasses all). The environment function defines the neighborhood path a phenotype representation needs to take in order to find the global optimum. It is independent of the genotype function. With this construction, the following performance measurements provide answers to the subsequent question. 1) For a given fitness function, if an MBOA can model the complexity function such that it changes accurate apply variation to the phenotype representation, then following the gradients in the fitness function will be no more challenging that if the complexity function was the identity. Given an environment function that is easy for an MBOA to overcome using a simple complexity function, a change to the complexity function, given the same environment function shows that the MBOA fails to model the variation operator. The reason is due to the complexity of the variation operator only. The algorithm will make random variations to the phenotype representation and therefore require an exhaustive search to find the global optimum. This enable the investigation of pairwise independent variations, overlapping variations and non-linearly representable variations. 2) For a given variation operator, if an MBOA cannot follow the gradient of the fitness function then the MBOA will fail independent on the complexity function used. Given a complexity function that is easy for an MBOA to learn and find a global optimal solution in a simple environment function, a change to the environment function, given the same complexity function shows that the MBOA fails to exploit the information in the model efficiently to follow the fitness gradient. The reason for failure is due to the inability of the MBOA to use the model in the way defined by the environment function only. This enables the investigation of population vs local search With this clear independent control of problem challenges this framework allows for the investigation of all the hypothesis proposed to differentiate SA-MBOAs using a single theoretical problem. This separation ensures that measured performance can be attributed to the specific problem challenges related to the type of variation to be applied or the fitness path. The problem challenges induced by the complexity function relate to the type of variation that must be applied, which is overcome by learning and representing sub-functions. The problem challenges induced by the environment function relate to the trajectory of the search, which is overcome by how model is used to improve a solution. Further, due to the same problem construction, accidental induce complexities will be consistent throughout all experiments therefore providing a meaningful relative performance between different problem challenges. The theoretical problem must present the correct problem challenges that are desired by the researcher to the optimizer. Further, this complexity must be proportionally controlled. For this theoretical problem complexity scales proportional to the size of the problem. This is achieved using building-blocks of controlled complexity. The building-block is defined to contain the desired problem challenge. Therefore, if an algorithm is unable to overcome the challenge presented by a single building-block, then more of the same will create an exponential search relative to the number of building-blocks (and therefore the size of the problem). An MBOAs that can successfully overcome the problem challenge will have a polynomial scaling. Finally, due to the environment function being easy to follow. The Phenotype representation for the global optimum is easily determined. In fact, for all environment functions investigated in this thesis a global solution has a phenotype representation of all 1's. Therefore, the solution representation can be easily calculated by inversing the complexity function. As the complexity function is a learnable function (that becomes clearer in section [complexity function], the global solution can be computed in linear time. Therefore, for all experiments performed using the theoretical problem in this thesis, the global solution is known.

5.3.2 Complexity Function

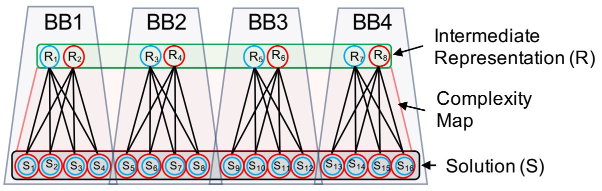

BB1
BB2
BB3
BB4
Intermediate Representation (R)
Complexity Map
Solution (S)
R1 R2 R3 R4 R5 R6 R7 R8
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16

FIGURE 5.2: The complexity function is a concatenation of sub-functions. Each building-block takes 4 arguments and produces 2 outputs. The outputs are used by the fitness function. The optimizer only has access to the arguments.

The complexity function defines the complexity of the variation operator an MBOA must learn and represent in order to follow the fitness gradient easily. A MBOA can only apply variation to the solution representation. Therefore, in-order to move efficiently at the intermediate representation, it must induce the relationships in the complexity function. To change the value at the intermediate representation requires a change to a control variable and a corresponding change to the dependent variable. To ensure that the correct variation operator is learnt, all states must be used during the search for a global optimum – the variation operator cannot be simplified or bypassed. Therefore, overlapping sub-functions must be able to be simultaneously satisfied. To ensure an MBOA must learn the complexity function, the complexity function is constructed from explicitly defined building-blocks. Each building-block contains a set of valid states (partial-solutions with respect to the global solution). All states not belonging to the valid set have inferior fitness. Therefore, a variation operator must accurately manoeuvre between the valid states in order to avoid deleterious fitness changes. Otherwise the fitness gradient cannot be efficiently searched and will instead require an exhaustive search to produce the correct assignment. A building-block represents a sub-set of partial-solutions, called states. The relationships between these states determines the complexity of the relationships a model must learn and represent. When there is only a single state to a building-block, there is no relationships to learn. When there are two states, the greatest complexity is pairwise. With more than 2 states, it is possible to introduce high-order relationships between states. In all building-block complexities explored in this thesis, a total of four states or size 4 is sufficient to create the desired problem challenges. The representation of a state of a building-block is uniquely represented at the intermediate representation. To investigate overlapping dependencies, at least two sub-functions must overlap. For this thesis it was sufficient only use two sub-functions in a building-block to explore overlapping dependencies. Each sub-function outputs a representation value of the state. Thus, the building-block takes 4 binary arguments and produces 2 binary outputs. The building-block can be interpreted as a multi-output Boolean function. Each sub-function constrains the set of possible assignments and together identify the state of the building-block.

A sub-function sets the dependency between two sets of variables, control (C) and dependent (D) variables, using a logical function. A control variable identifies the state of a sub-function and is used as the representation value at the intermediate representation. A dependent variable identifies the variables that have relationships with the control variable, but are not used to identify the state. Control and dependent variables represent the arguments to a sub-function. These are either linear, a linear representation of a single argument, or nonlinear a nonlinear representation of two or more arguments. The assignment of the dependent variable depends on the value of the control variables and the dependency function (f). If the dependent variable satisfies the dependency, as shown in equation 2, the sub-function is satisfied and a representation of the sub-function is made to the intermediate representation.

$$D = E(R) = E(C(S)) \tag{5.2}$$

Therefore, the complexity to make a variation to the intermediate representation requires a variation to the corresponding dependent variables. Overlap is induced when dependent variables share common control variables (or arguments in the case of nonlinear control variables). When the sub-function is not satisfied. The intermediate representation is a null value. The null value invalidates any fitness contribution from this assignment. The use of the null value ensures that only the sub-set of states to a building-block can contribute to the fitness function. Therefore, if the state of a building-block is valid, a variation operator can only move within the valid sub-set of states. Any variation that produces a state that does not belong to the sub-set will incur a deleterious fitness and therefore will get rejected. A building-block therefore takes variables from the solution representation and transforms it into 2 representation units. This construction is not limited to this building-block structure. However, it was the minimum size to induce the necessary problem challenges. The complexity function is then constructed as a concatenation of building-blocks. This is illustrated in Figure 2. The representation is therefore calculated as follows:

$$R = C[S] = [bb_1(s_1), \ldots, bb_i(s_i), \ldots, bb_m(s_m)] \tag{5.3}$$

Where R is output representation of the complexity function. bbi is an individual building-block containing sub-functions. M is the number of building-blocks in the complexity function and s is the sub-set of variables in S that are arguments for the building-blocks. In all problems used throughout this thesis, within a problem instance, bb is the same building-block, s is therefore the same size for all c which is 4. An essential characteristic for any benchmark to be considered useful in evaluating the performance of an optimisation algorithm is that it has proportionate control of complexity. Using this construction, the amount of complexity in the problem is controlled by increasing the size of the problem. By adding more of the same, an algorithm that fails to deal with the problem challenges presented by a single building-block will show an exponential scaling as the number building-blocks increases. The type of complexity is changed within a single building-block, unlike other theoretical problems that investigate overlap citekauffman1993origins, sherrington1975solvable, wang2013design. Therefore, evaluating the performance relative to the type complexity is quantified by the difference between scaling using different building-blocks. To investigate different problem challenges associated with the complexity of the variation operators, the relationship is varied. The different relationships investigated in this thesis are, pairwise, linear overlap, nonlinear overlap, pairwise independent separation. The sub-functions that define a building block are illustrated in Figure 1. C represents the control variable. C is also the representation of a sub-function if the sub-function is satisfied. D represents the dependent variable. The assignment of D must satisfy the relationship defined by f with variable(s) C. A single arrow indicates a direct representation of the arguments (linear). Two arrows indicate represent the correlation value between the two arguments.

5.3.2.1 Pairwise

Presented problem challenges: Linear transformation of variation operator

The pairwise relationship is used as a baseline complexity. All models are capable of modelling pairwise dependencies as verified by previous experiments cite pelikan2003hierarchical, goldman2015fast, hsu2015optimization. Therefore, using the pairwise building-block complexity provides a baseline performance for how an algorithm follow the fitness gradient. The pairwise function is constructed using two sub-functions that are non-overlapping. The relationship between the variables, f, is equivalence. The representation of a sub-function solution is a linear mapping of the state. This is sufficient to uniquely represent all solutions.

5.3.2.2 Cooperative Overlap

Presented problem challenges: Non-orthogonal variation Cooperative overlap is present when sub-functions that have shared variables (overlap) can be simultaneously satisfied and the values of the shared variables are defined by the context of all non-shared variables (the variables not overlapping). Cooperative overlap requires sub-functions that contain multiple assignments to satisfy an individual sub-function. A pairwise dependency does not provide this property. While a pairwise dependency, such as positively correlated, can provide multiple solutions, the assignment to a non-shared variable reduces the set of possible assignments to a single assignment. Instead, for cooperative overlap, the non-shared variable must only constrain the possible assignment of shared variable to a sub-set, rather than determine the assignment. Other sub-functions that share the variables then constrain the sub-set further to reduce the sub-set to a single assignment. In this case, the assignment for a sub-function solution must cooperate with the made to other sub-functions that is shares variables with. The problem challenges induced by this characteristic is a variation operator with overlapping linkage. Specifically, non-orthogonal variation (detailed in section [non-orthogonal variation]. For example a variation operation that requires variation to v1,v2 and v1,v3 and cannot be approximated by performing a variation decomposed variation of v1 then v2, or v1v2v3 then v2 for example.

Cooperative overlap differentiates from the other definitions. In agreement overlap, this definition is similar, however, the correct assignment to the shared variables can be determined from one of the non-shared variables, all other non-shared variables that are members of sub-functions that overlap become redundant. In conflicting overlap, the correct assignment is not dependent on the context of the non-shared values and doesn't require further variation once the conflict has been solved. This characteristic can be constructed using logic functions that take two arguments and produces a single output. In this case, the number of assignments that satisfy either 1 or 0 output of a logic function is not uniquely defined. Therefore, sub-functions that use the same arguments for separate dependencies have cooperative overlapping structure – A single sub-function does not constraint the arguments to a logic function. Two sub-functions are sufficient to uniquely constraint the arguments to a logic function.

Linear Cooperative Overlap Presented problem challenges: linear transformation of variation operator + Non-orthogonal variation. In the linear case, the relationship between the sub-function states can be approximate by a linear function. Specifically, the representation of state is a linear function of the arguments. This is achieved using the AND and OR logic functions between control and dependent variables. The AND function takes two inputs and produces a single output. The output assignment defines a sub-set of inputs that satisfy the function e.g. if the output is 1 then only inputs 11 satisfy the function. If the output is 0, the set of input 00, 01, 10 satisfy the function. When a second sub-function shares the same input variables, then the sub-set of assignments that satisfied the first sub-function is restricted further. Linear cooperative overlap is therefore created by using the AND as the relationships, f1, in SF1, and OR as the relationships, f2, in SF2 in Figure 3.

Nonlinear Cooperative Overlap Presented problem challenges: non-linear transformation of variation operator + Non-orthogonal variation. For the non-linear cases, the control variable represents the correlation between two arguments and not an argument as in the linear case. The values 1 and 0 represent a positive and negative correlation respectively. Overlap is present between the sub-functions as variable x1 is shared in the control representation. The sub-functions are the same as the pairwise independent Order 4 Odd case. However, by including a constraint that constrains the sub-set of valid solutions, removes the pairwise independency between sub-functions. In this case, the building-blocks can be separated using pairwise dependencies. The representation of relationships between variables remains non-linear non-the-less. The states to this building-block represent 1hot solutions. This complexity there shares similar characteristics for solutions in knapsack or discrete assignment problems encoded as a '1hot' representation. Therefore, a variation between assignment problems with constraints will share similar variation operators. To demonstrate the variation required for cooperative overlap, an example for the linear case is presented. In linear cooperative overlap two shared variables (x1,x2) and two non-shared variables (x3,x4) and logical AND and OR function are used for f_1 and f_2 respectively. Assignments that satisfy SF1 are 111*, 000*, 010* and 001* and assignments that satisfy SF2 are 11*1, 00*0, 10*1 and 01*1. There exsist four combinations that simultaneously satisfy both sub-functions, these are 0000 0101 1001 and 1111. Variation between solutions requires the following linkage structure (24,14,13,23,1234). To change the value at the intermediate representation requires a change to a control variable and a corresponding change to the dependent variable. For instance, to change R_1 (C_1), requires a change to D_1 only, D_2 only or D_1 and D_2 simultaneously. The correct variation is dependent on the context of D_1 and D_2. Therefore, the neighborhood presented at the intermediate representation is cooperative movements between sub-function states in order to maintain satisfaction of both sub-functions simultaneously

5.3.2.3 Pairwise Independence

Presented problem challenges: Pairwise independent separation of building-blocks + non-linear transformation of variation operator + non-orthogonal variation. Pairwise independent functions have been explored using parity functions. However, these can be easy to solve for a local search to change the representation of a building-block from odd to even is local using a single-bit variation. However, pairwise independence also occurs when the distribution of partial solutions has pairwise independent statistics. This type of relationships is constructed using nonlinear relationships between arguments. In this thesis, two types are explored. Order 3 odd parity and Order 4 odd parity. The order refers to the number of variables used and odd parity refers to all solutions within the order sum to an odd number. Therefore, all states are separated by 2 units of variation and are equidistant. Order 4 odd parity has the same relationships as the nonlinear cooperative overlapping problem. That is, the control and dependent variables represented correlations between arguments as depicted in Figure 3. However, the bias is removed. This produces an additional challenge. For each intermediate representation, there are two valid states. While this allows for many more global optima into the problem, it is never the less sufficient to evaluate the performances of SA-MBOAs and DO. What is significant about this complexity is that the representation is learnable using a shallow network. Order 3 odd parity is not representable using a shallow network. This is the only building-block complexity that requires a multi-level representation to compress the dimensionality of the search space. Order 3 uses only 3 out of the 4 arguments. To maintain the same size as all other building-blocks, argument 4 has a pairwise correlation with argument 3. As all MBOAs can represent the pairwise dependency between argument 3 and 4, this will induce not additional complexity to the problem.

A summary of each individual building-block function, bb_i, is summarized in Table1.

For all categories, local search will find a sub function solution easily; however, will not be able to vary between solutions, difference between all states are at least two units variations away - a variation of a higher-order organisation is required.

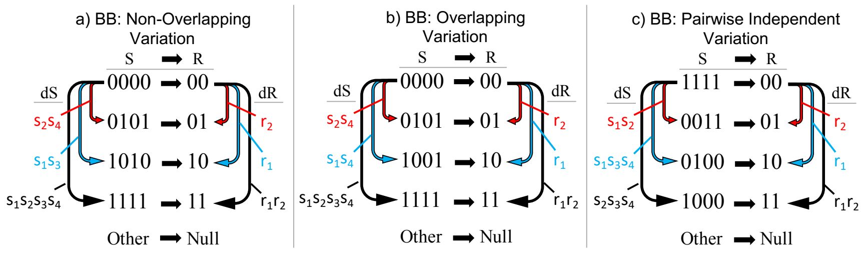
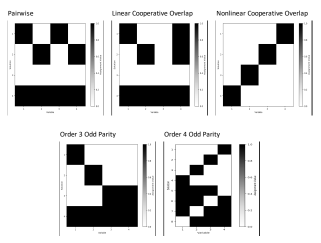
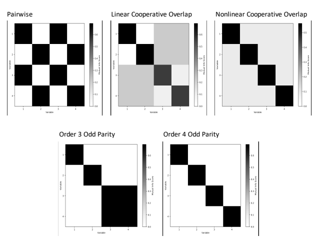


FIGURE 5.3: A BB of size 4 contains 4 solutions (S) represented by a binary representation R. All other solutions are represented by a null. A higher-order variation dS to S is required to make local variations dR in R. The types of higher-order variation explored are non-overlapping (a), overlapping (b) and pairwise independent (c).



(a) Set of valid solutions for each Building-block



(b) Pairwise Mutual Information for the set of valid solutions.

FIGURE 5.4: The subset of valid states to each building-block complexity and b) the mutual information between the subset of valid states to a building block.

Figure 5.4 presents the sub-set of valid solutions to a building-block. All other assignment combinations that are not part of the set are invalid and have a null representation. Figure 5 presents the pairwise mutual information the sub-set of valid solutions for each building-block complexity. The pairwise mutual information measures how much x_i determines the value of variable x_j, where i and j are argument index's. The observations are:

- Pairwise shows that x_1 determines the value of x_3 only and x_2 determines the value of x_4 only. Therefore, separable independent functions.
- Linear Cooperative overlap shows that x_1 and x_2 vary independently, but both x_1 and x_2 provide information for the value of x_3 and x_4.
- Nonlinear cooperative overlap shows that all variables provide some information to the assignment of a variable. Therefore, a nonlinear representation is required.
- Oder 3 and Order 4 Odd Parity shows that there is no pairwise statistical information. Note that for Order 3 Odd parity x_3 and x_4 are pairwise correlated only for consistency. Therefore learning using only pairwise statistics will fail to adapt beyond a single unit variation.

5.3.3 Environment Function

The environment function uses the intermediate representation of a solution to calculate the fitness. Therefore, the environment controls the problem challenge associated with the gradient to follow to improve a solution rather than type of variation to apply to a solution. The more accurate the model can represent the complexity function, C(S), the more efficient it can follow the fitness signal. All environment functions are simple (have polynomial time complexity) as proven by baseline cases using the pairwise complexity function. Further, if an algorithm cannot follow the gradient, it will get trapped an required an exponential search due to the non-additive decomposition. Thus all differences within the same environment are due to the suitability of a model to induce the structure in C() from a distribution of solutions. Different environments are explored that induce different problem challenges to an optimizer that are sufficient to differentiate all SA-MBOAs and DO. The environments explored in this thesis are: - Uniation: The problem challenge is to find a solution that has all the same states to a building-block. It is constructed such that all states can from part of the global optimum. - Non-additive hierarchical decomposition: The problem challenge is to rescale the variation operator to high-orders of organization. If an MBOA can represent C(), the rescaling is a linear function. - Nonlinear solution trajectory: The problem challenge is to perform a local-search that has a nonlinear path. If an MBOA can perform local search at the intermediate representation, to global optimum is reachable from any solution assignment. A non-additive function can either be constructed from a differential in the dependency strengths within and between sub-functions. In this case satisfying within sub-function dependencies are favored over satisfying between sub-function dependencies. However, a signal is always present for how sub-functions should combine, even if it has not been satisfied. The alternative is to restrict the visibility of dependencies between sub-functions unit the sub-function has been satisfied.Watson et al. (1998); Pelikan and Goldberg (2001). The latter construction is used here. This removes parameters associated with the number of dependencies and strength of dependencies to induce the same challenges as this can be an important relationship that effects the complexity of a problem Watson et al. (2011). Further, in natural problems, its intuitive to assume that dependencies between sub-functions is not observable until the immediate sub-function has been satisfied. One can only observe how to join two sub-functions when one has first found a solution to the sub-functions. For example: until we have the concept of an AND and OR logic gates we can't observe how these could interact and combine to form some higher-order function.

5.3.3.1 Compression Only

The simplest E mapping is one that does not contain higher level dependencies. Therefore a distribution of randomly generated solutions, that conserve the lower-order components, will provide the correct signal to follow to find the global solution, i.e., MIG and MIV can find the global solution. We refer to this mapping as compression only (CO) and forms our baseline case. The fitness (Equation 5.4), is calculated by separating R into two subsets R_1 and R_2 each containing a single representation unit from each BB (a BB has a representation unit in R_1 and R_2). The Hamming weight ($H()$) of R is taken and a fitness contribution calculated as the distance from the middle hamming weight of a subset ($m/2$, where m is the number of units in a subset), which has a fitness of zero. The contributions are summed to give the fitness of a solution. Using R_1 and R_2 provides four global optima, each one being a solution with all building-blocks containing the same PS,

$$F = F + |H(R_1) - \frac{m}{2}| + |H(R_2) - \frac{m}{2}| \quad . \tag{5.4}$$

Where F is the fitness of a solution, $abs()$ returns the magnitude of the value are r is the representation variable, and m is the number of building-blocks in a solution. The complete problem structure is presented in Figure 5. The constructing is illustrated in Figure 6. The gradient to follow is determined by the majority in the solution. To allow for all states of a building-block to form a global solution, the representation is separated into two modules that separate each representation of a building-block. Each module sums the representation and assigns a fitness. Therefore, variation between states must be accurate and conserve over sub-functions. The optimal solution for each module is either a representation with all zeros or all ones. Therefore, the optimum for each sum module provide a combination of 4 global solutions, which represents a solution that for every building-block has the same building-block state. This function is simple for all MBOAs and provides the baseline for the different environment functions explored.

The time complexity to find a global solution is linear with respect to the intermediate representation size as every variation made to a representation variable will either already be correct or be varied to the correct assignment. The linear gradient provides a clear signal to accept variations that increase that similarity between building-block states. This is exploitable if the can vary between all states to all sub-functions. If an MBOA fails, it simply cannot search between states of a building-block.

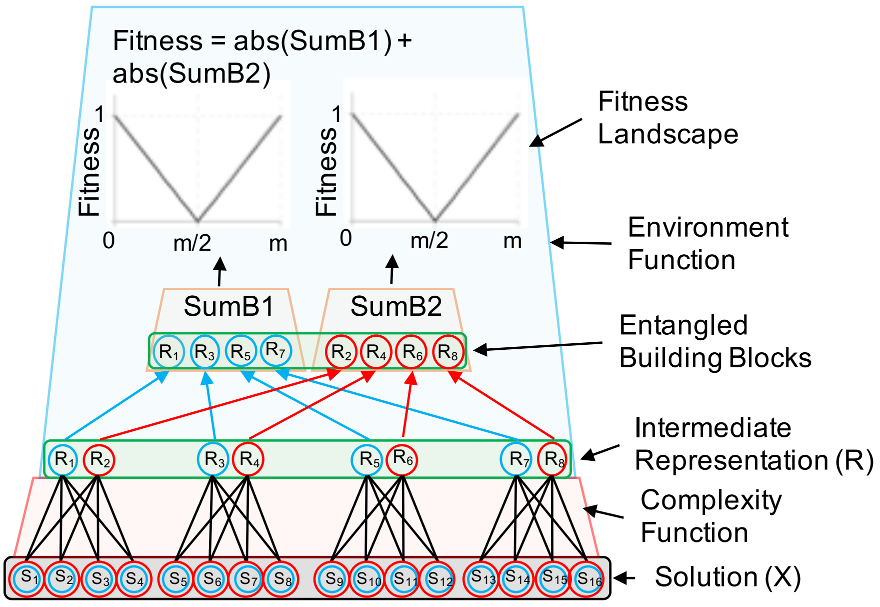
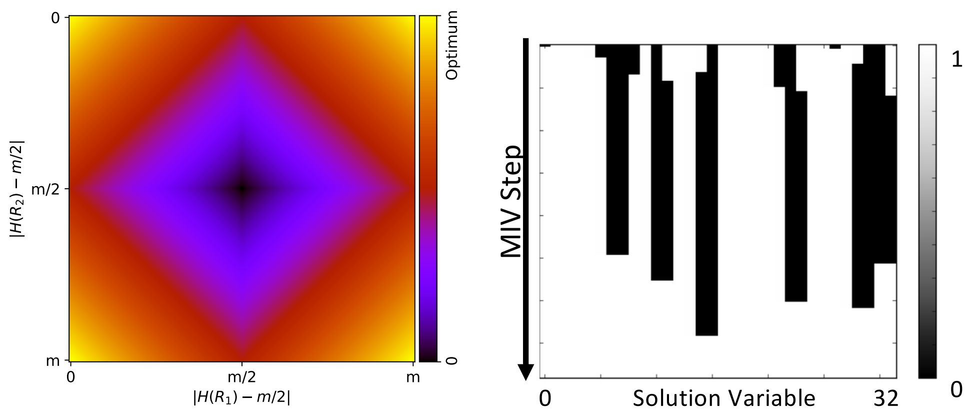


FIGURE 5.5: Problem structure for the modular theoretical optimisation problem. If an optimiser can separate the building-block solutions accurately, the problem is a simple combinatorial search with a clear gradient to the global optimum

Figure 5.6(a) represents the fitness landscape relative to the higher-level representation of a solution for the Combination Only environment ($E = CO$). The landscape contains four globally optimal solutions. The correct variation to apply to a solution is dependent on the most common solution to all building-blocks in that solution. Each solution can contain a different common building-block solution. Therefore the correct variation to apply is dependent on the solution's state. An algorithm must therefore conserve all building-block solutions to efficiently find a globally optimal solution.

5.3.3.2 Hierarchical

Interesting cases for MBOAs arise when a subset of low-order components can form high-order components. Hierarchy is one example where higher-order components are constructed out of nested lower order components. As such, hierarchy requires a recursive compression of the neighbourhood that additively combines lower-order components to higher-orders of organization in-order to navigate the fractal landscape Watson et al. (1998). We use the same hierarchical construction here as all MBOAs can successful overcoming the challenge. The hierarchical dependencies are represented by a binary



(a) The fitness landscape for the combination only environment ($E = CO$) relative to the intermediate higher-level representation R.

(b) Solution Trajectory to Global Optimum

FIGURE 5.6: The Fitness landscape (a) and solution trajectory (b) required to find the global optimum solution.

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 5.5. A parent node with a non-null value is assigned a fitness benefit. The total fitness, Equation 5.6, is the sum of all fitness contributions from all parent nodes,

$$r^i_{l+1} = A(r^i_l, r^j_l), \quad A(r_i, r_j) = \begin{cases} r_i, & \text{if } r_i = r_j \\ \text{null}, & \text{otherwise} \end{cases}, \tag{5.5}$$

$$F = F + \sum_{l=0}^{D} \sum_{i=0}^{2^{D-l}} f(r^i_l), \quad f(r_i) = \begin{cases} 0, & \text{if } r_i = \text{null} \\ 1, & \text{otherwise} \end{cases}. \tag{5.6}$$

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

The hierarchical environment is a non-additive decomposable structure. Hierarchy presents a strict nesting of sub-functions. The nesting combines lower-level sub-functions into higher-orders of organization. Hierarchy presents the challenge of recursively rescaling the variation to high-orders of organisation. This rescaling requires correctly decomposing the problem, representing the solutions to building-blocks and maintaining a diversity of the solutions such that partial solutions are not lost during selection Pelikan and Goldberg (2003b, 2006). The hierarchical fitness function is calculated from a multilevel representation of the solution. The multilevel representation is calculated using the following equation:

$_h(f_(h-1)^m(S), f_(h-1)^(m+1)(S))$ Where f represents the equivalent function that has internal dependencies and produces an output based on the satisfaction of those dependencies, h represents the hierarchical depth of the function, S represents the variables

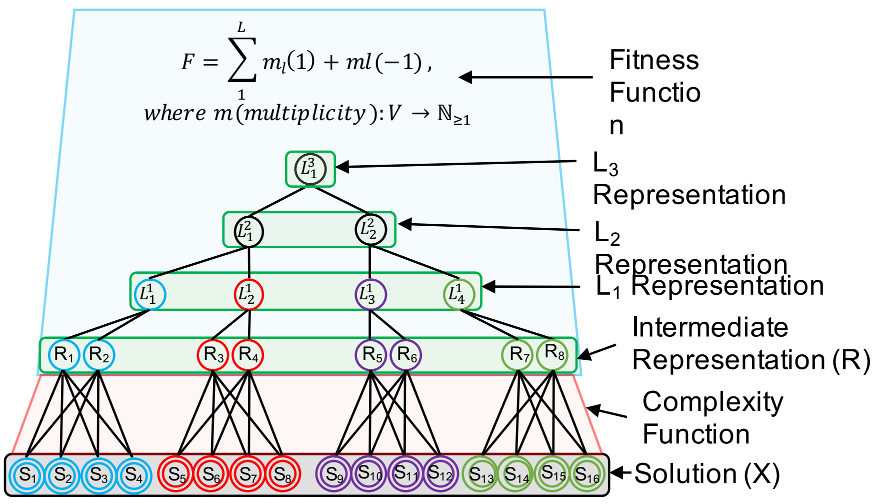

$F = \sum_{1}^{L} m_l(1) + ml(-1)$,
where $m(multiplicity): V \to \mathbb{N}_{\geq 1}$
Fitness Functio n
L_3 Representation
L_2 Representation
L_1 Representation
Intermediate Representation (R)
Complexity Function
Solution (X)
L_1^3
L_1^2 L_2^2
L_1^1 L_2^1 L_3^1 L_4^1
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12} S_{13} S_{14} S_{15} S_{16}

FIGURE 5.7: Hierarchical Problem Construction

used as arguments to the function m represents the type of function used (it is not a requirement that all functions are the same). This construction has been used for previous benchmark problems containing hierarchy Watson et al. (1998); Pelikan and Goldberg (2001). The environment used in this construction is the same as the Hierarchical If and only If problem. All MBOAs are capable of solving this problem structure using an identity complexity function Pelikan et al. (2003); Goldman and Punch (2015); Hsu and Yu (2015).

Further, this is the same example for bivariate subfunction. The hierarchical fitness function is a nesting of bivariate subfunctions where a non-null control variable is awarded a fitness point proportional to the level of the non-null control variable, specifically 2^h. The fitness function is an additive combination of all non-null control variables in the nested tree of the subfunctions.

This construction induces increasingly larger areas of neutrality as the representation level increases. These space of neutrality are can be navigated by search at the lower-level representation. This creates a fractal fitness landscape that requires, for each level of representation, a re-scaling of the variation operator to a high-order of organization such that neighborhood is adapted to remove the neutrality at the next level. In doing so, satisfying the next level sub-functions easy. Due to the problem challenge or re-scaling the variation operator, an accurate representation of the structure is required.

Hierarchy presents the challenge of re-scale the variation to high-orders of organisation this requires both correctly decomposing the problem and representing the partial solutions efficiently and maintaining a diversity of the solutions such that partial solutions are not lost during selection if they are yet to be correctly combined Pelikan and Goldberg (2003b, 2006). The hierarchical if and only if function is used as the fitness function.

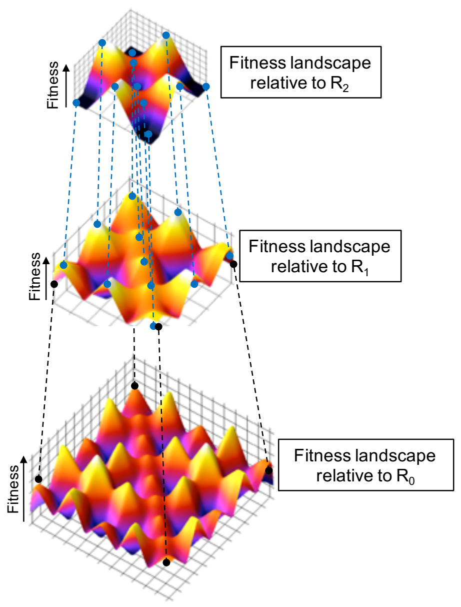
Each node represents an additive compression of its child nodes. As in the case of the complexity function construction, the dependencies between sub-functions at high levels is only visible when the sub-functions are satisfied. The dependency between the child nodes is equivalence. If two child nodes have the value, a fitness reward proportional to 2^l, where l is the hierarchical level of the node, is awarded to the parent node. the child nodes have the same value. The value of a node is the same as a child node if the child nodes agree, otherwise it is a null. An algorithm must be capable of representing the hierarchical decomposition effectively to follow the fitness gradient.

In the case of an approximated variation, if the variation to the original sub-function was to the correct solution, the disruption it has caused by not cooperating with the other sub-functions causes a net fitness loss. Therefore, the previous solution is fitter and is selected. An algorithm using an approximation of the overlap requires an exponential exploration of the possible combinations to satisfy the between sub-function dependencies. Therefore, an algorithm must accurately represent the overlapping sub-functions for effective search. If an algorithm cannot, and variation disrupts the solution for the overlapping sub-function, then the variation is generally deleterious. The optimizer must then preserve the solutions and search in combination of sub-functions to satisfy higher-order dependencies. In doing so we satisfy the challenge of causing variation of building-blocks while maintain the dependency with other building-block that share the same control variables. We demonstrate this is section **??**.

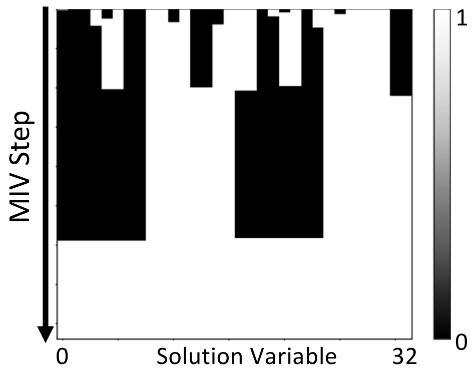
Figure **??** represents the fitness landscape relative to the higher-level representation of a solution for the Hierarchical environment ($E = H$). The landscape has multiple levels of representation, each a compression of the landscape from the layer below. Specifically, a variation operator that enables movement between the peaks at the 1st level representation observes the landscape at the representation above. In this hierarchical landscape, higher-level representations also contain relative modality and therefore cannot be navigated using a variation operator appropriate for the lower levels. An optimiser must therefore re-scale the variation operator to move in the higher-level landscapes. The representation is a simple additive compression, therefore, the algorithm must additively combine a pair of variation operators used at the layer below.

Entangled Sub-functions

The general construction of hierarchical function is a nested function $h(f(x), g(y)$. HiFF is described as containing strong non-linear dependencies between variables. However, the problem construction can be approximated using linear dependencies. This is due to the dependencies within a sub-function. For example. at level one the building blocks are of size two, with solution 11 and 00. The next level combines these solutions to form a module of size 4 with solutions 1111 and 0000. Thus the information of the low-level decomposition is no longer required. This is clearer when we look at the algebraic expression for HIFF $h1 = h(f(x1, x3), f(x2, x4)$ and $h2 = h(f(x5, x7), f(x6, x8)$. Variables


Fitness landscape relative to R_2
Fitness landscape relative to R_1
Fitness landscape relative to R_0
Fitness
Fitness
Fitness

(a) An illustration of the fitness landscape for the hierarchical environment ($E = H$) relative to the intermediate higher-level representation R. The Hierarchical landscape consists of multiple levels of representation. Each layer represents the fitness differences between the peaks in the layer below.



(b) Solution Trajectory to Global Optimum

FIGURE 5.8: The Fitness landscape (a) and solution trajectory (b) required to find the global optimum solution.

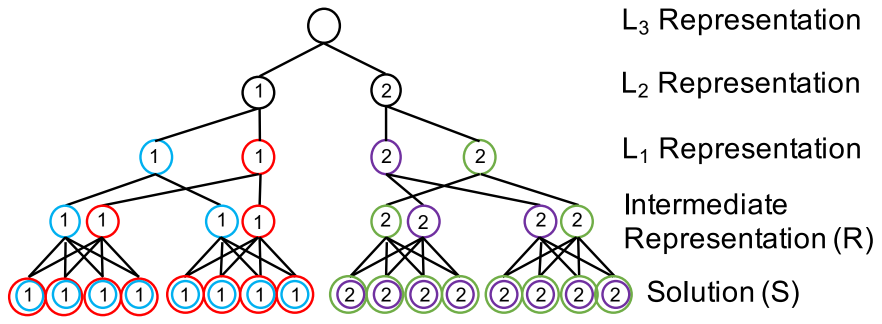


FIGURE 5.9: Hierarchical optimisation problem with entangled sub-functions up to depth 2. At depth 3 all sub-functions are separate. Below depth 2 sub-functions are entangled where a individual input variation is an argument for multiple sub-functions on the same level.

are not shared at higher-level functions. Thus the problem can be approximated using linear dependencies between variables, once a high-level solutions has been found, the low-level sub-functions can be combined using a pairwise dependency between the variables. As such there is no requirement for a deep model to perform the dimensionlaity reduction for this problem. As we explore later. To introduce non-linear dimensinolaoty reduction, multi-variate dependencies that transfer through all level of the hierarchical structure. This is achieved using cooperative overlapping functions. Here, when shuffled, $h1 = h(f(x1, x2, x3), f(x5, x6, x7)$ and $h2 = h(f(x1, x2, x4), f(x5, x6, x8)$. In this cases variables are (x1,x2,x5,x6) shared between sub-functions at the high-level. Therefore, the representation must allow for this separation, which is achieved by representing the separation of functions of the lower level. In DO this is achieved using a deep network.

To validate this further, the depth of the shared variables can be controlled by using a pairwise compression in the hierarchy. If the compression is limited to a single building block, then the solution to the level 1 function will be $h1 = h(f(x1, x2, x3), f(x1, x2, x4)$ and $h2 = h(f(x5, x6, x7), f(x5, x6, x8)$. In this case, no function at the first level shares variables with other functions at the same level. Therefore, the structure has been reduced back to a linear compression, for what a shallow model can solve, and problem challenge of overlap has been removed. The problem has been reduced to the complexity of HIFF. If the 1st layer is constrained to sharing variable between two-building blocks, then by the second level function, functions will have no shared variables and so forth. Figure 1 provides a clear illustration of this control of problem depth.

Hierarchy doesn't create a deep problem alone. It is also required that the dimensional reduction created by the hierarchy is non-linear. In hierarchical problems, once sub-solutions are correctly combined, there is no requirement to preserve partial solutions that enabled this combination (one we have the correct solutions we can move on to the next level). Therefore if the dimensional reduction is linear, information can be captured by a shallow model.

The hierarchical construction is tree structure which inherently contains no overlap. However when the leaf nodes contain overlapping dependencies, as long as nodes does not combine two overlapping sub-functions, the overlapping dependencies will be present at the corresponding hierarchical level. This therefore provides us with a method to evaluate the performance with regards to the depth at which overlapping dependencies occur. Moreover, as the level of hierarchy increases, the number of overlapping dependencies that must be dealt with also increases. Concretely, at higher levels of organisation, satisfying a deep node will require a variation that does not disrupt 2^depth number of sub-functions, all of which could share some of the variable. The correct variation must therefore be in the context of multiple sub-functions. This is different to the shallow version, where the variation is only required in the context of only one sub-function.

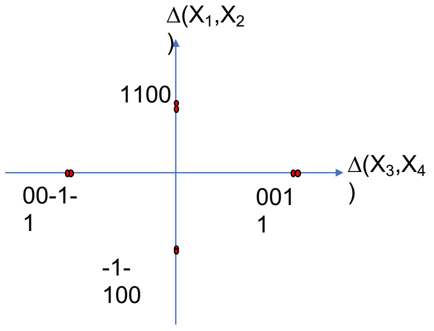
It is possible to control the depth of overlap controlling the linkage information within sub-trees of the hierarchical tree. Specifically, by causing random linkage in a sub-tree with root node at depth d, where $d <= h$, at $h > d$ there will be no overlapping dependencies present in the problem. Figure **??** presents an example of deep problem structure with $d = 2$. Each node on the bottom layer is a problem variable, and each node on a higher level is a latent problem variable. The lines between nodes represent the linkage between sub-functions. At $h = 2$, there are 4 sub-functions, colour coded to identify dependent solution variables of which a variable contributes to multiple sub-functions at $h = 2$. At $h = 3$, there are 2 sub-functions numbered to identify dependent solution variables of which a variable contributes only to a single sub-function at $h = 3$.

The hierarchical construction is a tree structure which inherently contains no overlap. However, when the leaf nodes contain overlapping dependencies, then as long as nodes do not combine two overlapping sub-functions, the overlapping dependencies will be present at the corresponding hierarchical level. It is possible to control the depth of overlap by controlling the linkage information within the sub-trees of the hierarchical tree. Specifically, by causing random linkage in a sub-tree with the root node at depth d, where $d <= h$, at $h > d$ there will be no overlapping dependencies present in the problem. Figure **??** presents an example of deep problem structure with $d = 2$. Each node on the bottom layer is a problem variable, and each node on a higher level is a latent problem variable. The lines between nodes represent the linkage between sub-functions. At $h = 2$, there are 4 sub-functions, colour coded to identify dependent solution variables of which a variable contributes to multiple sub-functions at $h = 2$. At $h = 3$, there are 2 sub-functions numbered to identify dependent solution variables of which a variable contributes only to a single sub-function at $h = 3$. Therefore the dependencies between pairs of sub-functions is reduced to bi-variate, regardless if it was a linear or nonlinear overlapping dependency defined at the lowest level.

The ability to control the depth of overlap enables to evaluate the performance of an algorithm to deal with conserving an increasing number of sub-function solutions. As the level of hierarchy increases, the number of overlapping dependencies that must be dealt with also increases. Concretely, at higher levels of an organisation, satisfying a deep node will require a variation that does not disrupt $2^d epth$ number of sub-functions. The correct variation must, therefore, be in the context of multiple sub-functions, with the number of sub-function doubling as the depth of overlap is increased. This challenge is substantially different from the shallow version, where the variation is only required in the context of only one sub-function. We use this control of complexity to gain greater insight into the performance of MBOAs. This allows for evaluating the performance of an algorithm as the complexity of the depth of the overlap increase. An algorithm that can deal with overlapping structures, should be insensitive to the depth of overlap – how much overlap is present – the size of overlap. Their exists no benchmark optimisation problems that contain hierarchy and overlap

5.3.3.3 Cyclic Path

The final challenge we explore, is when higher-order components are not a combination of lower-order components, but rather, they are found by navigating in the space of lower-order components. We achieve this by creating a unique path to the global solution. The



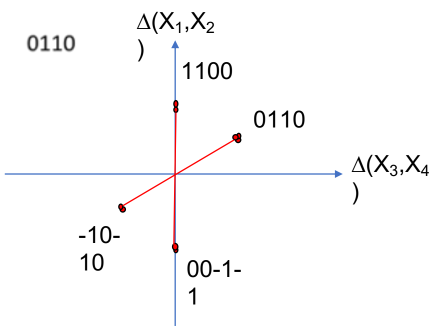
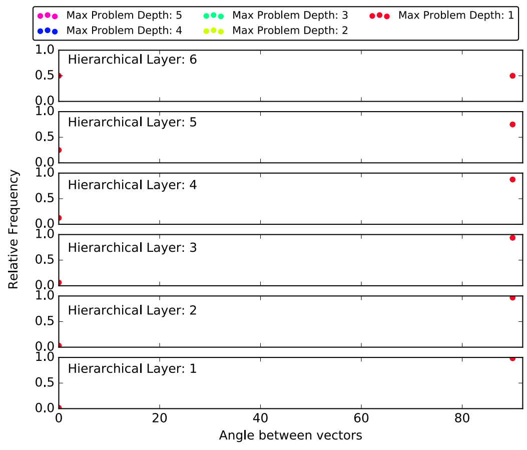
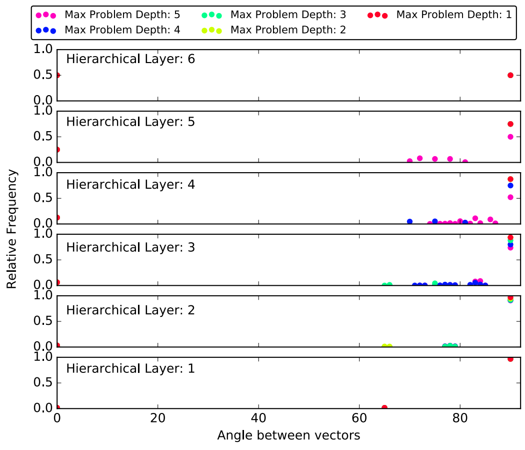


FIGURE 5.10: Orthogonal (a) and non-orthogonal variation (b)



(a) Angle between variation operator vectors for $C = NO$



(b) Angle between variation operator vectors for $C = LO$

FIGURE 5.11: Non-Orthogonal variation

R representation is split into two subsets R_1 and R_2, each containing a single representation unit for each BB. The Hamming weight ($H()$) of each subset is calculated and used as coordinates for a 2D fitness mapping. 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 as $(m, 0) \rightarrow (0{,}0) \rightarrow (0, m) \rightarrow (m, m)$, where (m, m) is the global optimum. We refer to this E mapping as the cyclic path (CP) due to a representation units cycling between assignments to find the unique higher-order component,

The fitness function for the nonlinear path problem is defined in Equation.

$$F = F + CP[H(R_1), H(R_2)] \quad . \tag{5.7}$$

Importantly, the path length scales polynomially with respect to the problem size and is easy to follow by performing local search in R, thus differentiating it from a long path problem Horn et al. (1994).

The problem challenges presented to an algorithm is the ability to explore the neighborhood of search space using the model. The complete problem structure is presented in Figure 8. The nonlinear path environment function is a simple landscape that requires following a specific solution trajectory to the global solution. It is analogous to the long path problem, however, in this case the path does not scale exponential with the problem size. The length of the path scales linearly. The challenge is the nonlinear trajectory where assignment to variables go between states multiple times to find the global optimum. The landscape is illustrated in Figure 7. The landscape is constructed from an array of fitness assignments where the indexes are calculated from modules SumB1 and SumB2 for the representation as demonstrated in Figure 8. As in the uniation environment, the representation is separated such that each module represents only a single sub-function of each building-block. Therefore, variation between states must be accurate and conserve over sub-functions

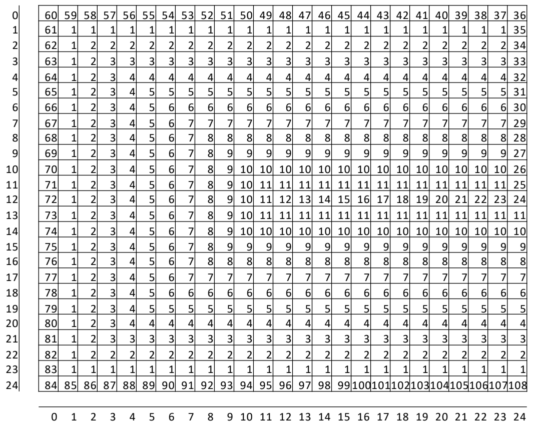
The landscape is constructed using two modules, the same as in the uniation problem. Fitness landscape however is changed. Each module represents the coordinates of a landscape. The landscape is illustrated in Figure 7. SumB1 represents the x-axis coordinates and SumB2 represents the y-axis coordinates. The coordinates define the location on the landscape. The landscape has a ridge around the perimeter that leads to the global solution at SumB1 = SumB2 = N (where N is the number of building-blocks in a problem. The global optimum is reachable from all possible intermediate representations. A random solution, that is not on the ridge, will be encourage towards the bottleneck before then following the path. The path requires the first half of representation to all 0's, while maintaining the 2nd half at all 1's. Then the 1st half is held at all zeros while the 2nd half is varied to all 0's. Then the 1st half is varied to all 1's while holding the 2nd half at all zeros. Then finally the 1st half is held will the 2nd half is varied to all 0's. Of course any variation that does not maintain the other half's representation will be deleterious. This trajectory is illustrated in Figure 9 which uses an identity complexity function. A local-search can easily find the global optimum.

Figure **??** represents the fitness landscape relative to the higher-level representation of a solution for the Cyclic Path environment ($E = CP$). The landscape has a unique path to the global optimal solution at coordinates (m, m). The path to reach the global optimum solution can be navigated by single-unit variations at R.

5.4 Summary

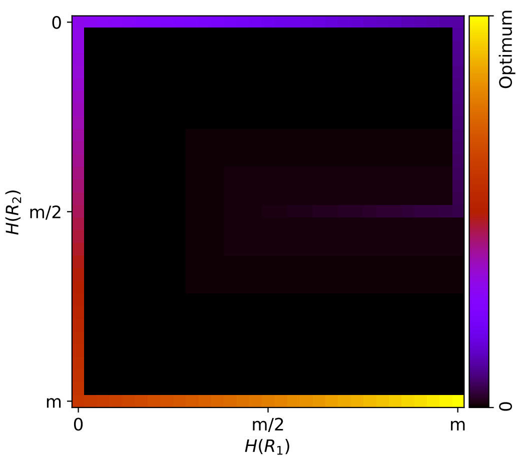
Identify the complexity and environment functions Cpair, Cl$_coop, CNL_coop, Cparity3, Cparity4EUni, E$

capable of independently exploring different complexities of dependency structures in a non-additive separable optimisation problem. This section constructed a theoretical



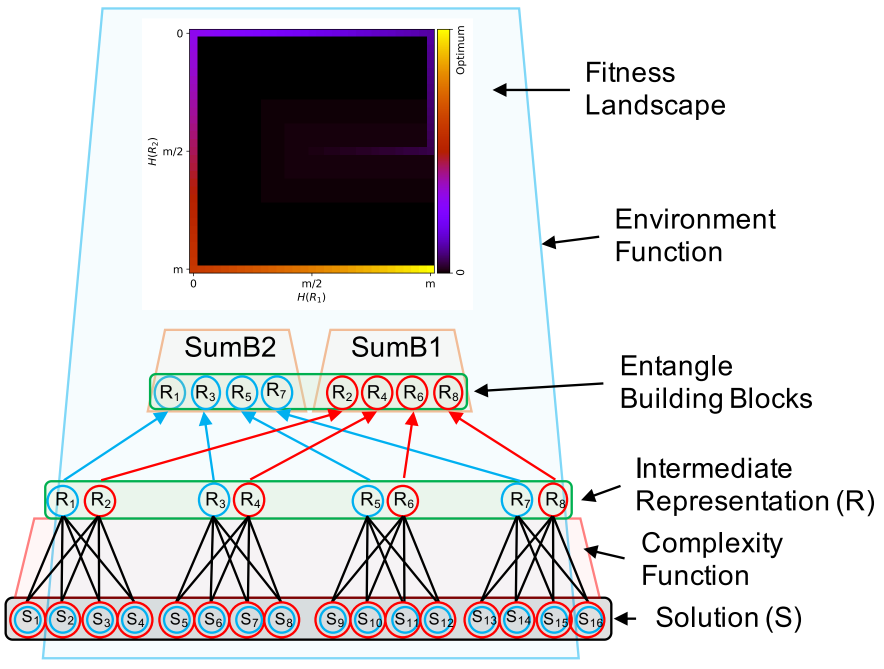
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36
1	61	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	35
2	62	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	34
3	63	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	33
4	64	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	32
5	65	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	31
6	66	1	2	3	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	30
7	67	1	2	3	4	5	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	29
8	68	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	28
9	69	1	2	3	4	5	6	7	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	27
10	70	1	2	3	4	5	6	7	8	9	10	10	10	10	10	10	10	10	10	10	10	10	10	10	26
11	71	1	2	3	4	5	6	7	8	9	10	11	11	11	11	11	11	11	11	11	11	11	11	11	25
12	72	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
13	73	1	2	3	4	5	6	7	8	9	10	11	11	11	11	11	11	11	11	11	11	11	11	11	11
14	74	1	2	3	4	5	6	7	8	9	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
15	75	1	2	3	4	5	6	7	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
16	76	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
17	77	1	2	3	4	5	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
18	78	1	2	3	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
19	79	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
20	80	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
21	81	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
22	82	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
23	83	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108

(a) Absolute Values in Landscape



(b) Fitness Landscape Visualised

FIGURE 5.12: Environment function for the Manifold fitness landscape. The input is separated into two modules. The sum of ones in each module is mapped to the function to provide the fitness of the solution. (a) The absolute values used for a representation size of 24. B) A zoomed (limited to 20 fitness points) example of the fitness landscape for the general case (N).



Fitness Landscape
Environment Function
SumB2
SumB1
Entangle Building Blocks
Intermediate Representation (R)
Complexity Function
Solution (S)
R_1 R_3 R_5 R_7 R_2 R_4 R_6 R_8
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12} S_{13} S_{14} S_{15} S_{16}
$H(R_2)$
$H(R_1)$
Optimum

FIGURE 5.13: Problem structure for the manifold theoretical optimisation problem. The gradient is simple to follow if an optimizer can model and separate the directions of variation and hill-climb

problem that facilitates evaluation of all problem challenges identified. Further, the construction allows for:

1. Independent evaluation of each problem characteristics.
2. clear structure to allow for efficient computation of the global optimum,

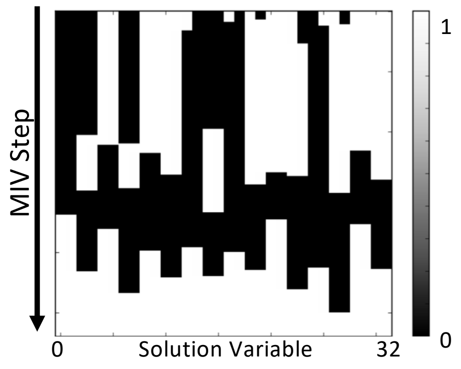


FIGURE 5.14: The solution trajectory required to find the global optimum solution.

3. clear problem challenges that must be over come by the optimiser.
4. Construction within the same framework so that comparison of an algorithms performance between different complexities can be clearly attributed to the change in problem complexity (rather than other induced complexity caused by the change in problem construction)
5. Proportional control of problem complexity to facilitate scailiblity analysis

The global fitness for NSP is calculable in linear time combines these characteristics in a clear and explicit controllable way. This facilitates easy calculation of the global optimum. The theoretical problems allow researchers to ask the following questions

Contributions - non-separable pairwise independent function at all depths - clarification on pairwise independent functions (using one hot) caused by a cooperative overlap of two sub-functions that are representable using a single bi-variate. - linked to multi modality as it is about the statistical appearance can cause pairwise independence, not just a parity function. - one hot example - is it rare, don't it, used in lots of binary representation of problems, easily imaginable in knapsack problems - maybe a reason for why linkage learning for mkp was poor using ltga NN are a method that do not make the assumption of pairwise construction and are therefore able to learn pairwise independent functions using a simple algorithm a clarification going from simplest - pairwise only to pairwise independent extremes, Presenting a continuum of problems.

- Multi-level representation – higher – order representation Further, by constructing the dependencies out of logic functions, we gain an understanding of the complexity of learning the structure. For instance, learning an XOR function is more complicated than learning a AND function.

5.5 Performance Comparison

In this Chapter, we aim to understand how the model used in state-of-the-art Model-Building Optimisation Algorithms (MBOAs) directly affects an algorithms ability to exploit structure in optimisation problems. Considering the model used has a direct impact on the ability to model the dependencies between variables and therefore represent partial solutions, it is surprising that results do not differentiate the performances between MBOAs beyond an efficiency difference. We aim to understand the problem characteristics that create a failure success differentiation between MBOAs that use different models.

This chapter aims to identify the problem challenges that differentiate the performances between the SA-MBOA. The results for DO are presented in Chapter 5. Results for rHN-g are included. Although not a state-of-the-art method, the way in which information is exploited from the model differs significantly from the SA-MBOAs. The evaluation of an MBOAs performance is measured by a scalability analysis for different complexity functions and different environment functions. In all cases, the number of function evaluations to first locate a global optimum is measured. The time complexity is omitted from the analysis. All SA-MBOAs have polynomial time complexity to construct and sample the model. When the number of functions evaluations required to find a global optimum solution scales exponentially, the time associated with constructing and sampling the model (given that it is polynomial) is of less concern. An algorithm is said to succeed or fail in dealing with the problem complexity if it scales polynomial or exponentially, respectively as the size increases. As discussed in Chapter 2, the use of local search to initialise a population of an MBOA can significantly improve the performance of an MBOA. While this is often referred to as a hybrid MBOA, with the definition of MIV, this is not the case. Local search is MIV but with a model that represents the identity function. The research question is focussed on understanding the difference between what models can and cannot do. All algorithms use a local search to initialise the population, so that the distribution of solutions contain optimal solutions to the building-blocks. Therefore, the scaling behaviour is solely attributed to the what the model can represent and how this information is exploited rather than finding a good distribution of building-block solutions. For MBOAs that use a population, the optimal population size was found by an incremental search. The population started with a size that is insufficient to find a global optimum solution and incrementally increase by 10The bisection method, often used by other researchers was not a suitable for investigating hierarchical problems with overlapping structure. Due to the random shuffling, some instances can be easier than others, therefore the bisection method was not always a reliable method for finding the optimal population size. The parameters used for MBOAs were manually tuned using a sample of problem cases. No measurable differences were found from the default parameters. We omit to evaluate the computational cost in this paper. All MBOAs take polynomial time to construct and sample the model. Parameter tuning and population size tuning add to the complexity of an MBOA. However, this is not the aim of this thesis. The aim is to understand the performance differences between the models capacity and how the model is used in SA-MBOAs. Therefore, by tuning the population size for each instance, we measure the performance of the model and not the design decision or implementation details of the algorithm.

This section presents the scalability of all SA-MBOA for all complexity functions and environment functions

5.6 SOTA MBOAs vs DO

5.6.1 Model: Representation

In this section, the Uniation environment function is used to evaluate the performance of an MBOA to represented the complexity function.

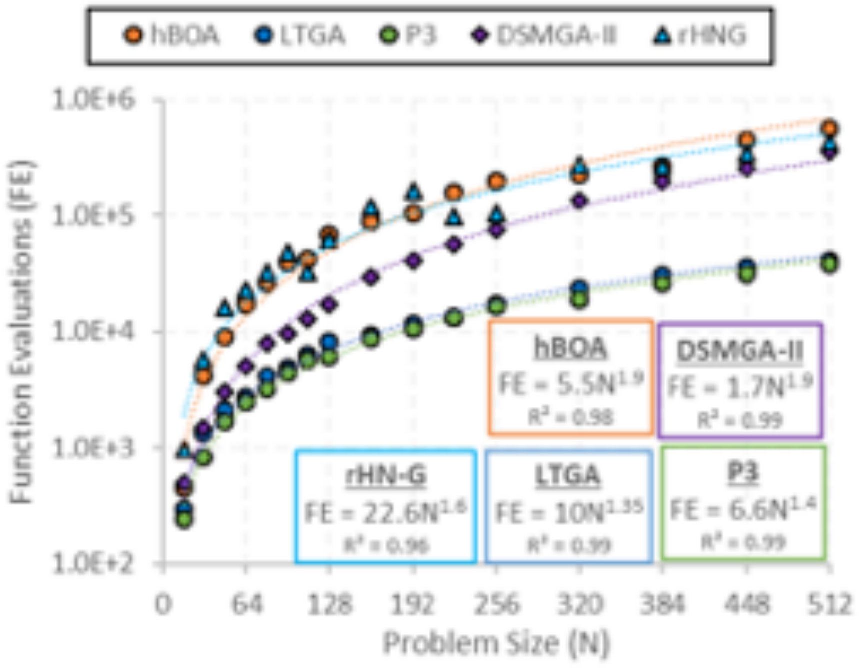
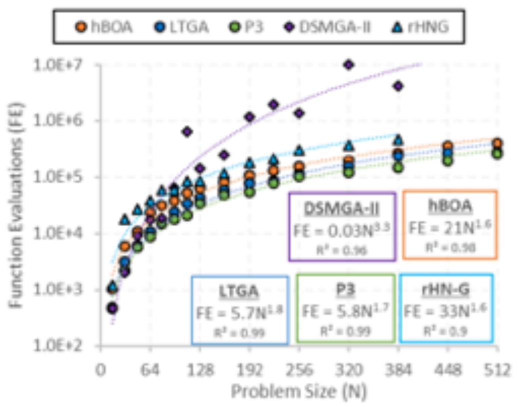


FIGURE 5.15: Scalability performance of all SA-MBOAs and rHN-G for the $C_{pair}E_{co}$ problem. All algorithms are successful in representing pairwise dependencies

Figure Figure 5.21 presents the scalability results for all SA-MBOAs and rHN-G for the CpairEUni problem. All algorithms have a polynomial scaling. Therefore, verifying that all models are capable of accurately representing the pairwise relationships.

Figure Figure 5.16(a) presents the scalability results for all SA-MBOAs and rHN-G for the $C_{lcoop}E_{co}$ problem. All algorithms have a polynomial scaling. DSMGA-II shows a significant change in the performance compared to the pairwise complexity function.



MBOA_Images/ExperimentalResults/Cnlcoop_Eco.pdf

(a) Scalability performance of all SA-MBOAs and rHN-G for the $C_{lcoop}E_{co}$ problem. All algorithms are successful in representing pairwise dependencies

(b) Scalability performance of all SA-MBOAs and rHN-G for the $C_{nlcoop}E_{co}$ problem. All algorithms are successful in representing pairwise dependencies

FIGURE 5.16: Scalability performance of all SA-MBOAs and rHN-G for the $C_{lcoop}E_{co}$ (a) and$C_{nlcoop}E_{co}$ (b) problem. All algorithms are successful in representing pairwise dependencies

Both LTGA and P3 also show an increase in the scaling that is more in line with the performance of hBOA and rHN-G. hBOA remained insensitive to change in complexity function from pairwise to linear cooperative overlap.

Figure Figure 5.16(b) presents the scalability results for all SA-MBOAs and rHN-G for the $C_{nlcoop}E_{co}$ problem. rNH-G fails to represent the problem structure due to using a correlation matrix as the model. LTGA also fails to represent the problem structure. Whereas P3, an algorithm that uses multiple linkage-tree models, is successful. P3, hBOA and DSMGA-II all show a polynomial. P3 and hBOA show an increase in the exponent when compared to the linear cooperative case.

Figure Figure 5.17(b) and Figure Figure 5.17(b) presents the scalability results for all SA-MBOAs and rHN-G for the $C_{3odd}E_{co}$ and $C_{4odd}E_{co}$ problem respectively. P3 is the only algorithm that doesn't scale exponential for both problems. It is hypothesis that this is due to multi-level representation of the search distribution. This is explored further in the discussion. While the pairwise independence wasn't sufficient to cause exponential scaling for all MBOAs, the scaling performance of P3 is poor non-the-less.

5.6.2 Model: Exploitation

In this section, the cyclic path environment function is used to evaluate the performance of an MBOA to exploit information from the model.

Figure Figure 5.18(a) and Figure Figure 5.18(b)

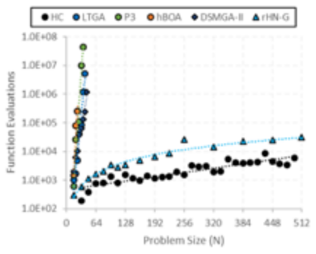
MBOA_Images/ExperimentalResults/C3odd_Eco.pdf

(a) Scalability performance of all SA-MBOAs and rHN-G for the $C_{3odd}E_{co}$ problem. All algorithms are successful in representing pairwise dependencies

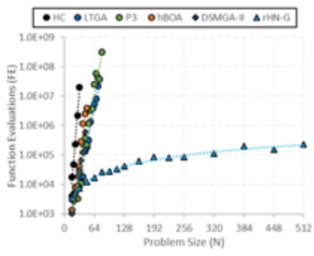
MBOA_Images/ExperimentalResults/C4odd_Eco.pdf

(b) Scalability performance of all SA-MBOAs and rHN-G for the $C_{4odd}E_{co}$ problem. All algorithms are successful in representing pairwise dependencies

FIGURE 5.17: Scalability performance of all SA-MBOAs and rHN-G for the $C_{3odd}E_{co}$ (a) and $C_{4odd}E_{co}$ (b) problem. All algorithms are successful in representing pairwise dependencies



(a) Scalability performance of all SA-MBOAs and rHN-G for the $C_{identity}E_{cp}$ problem. All algorithms are successful in representing pairwise dependencies



(b) Scalability performance of all SA-MBOAs and rHN-G for the $C_{pairwise}E_{cp}$ problem. All algorithms are successful in representing pairwise dependencies

FIGURE 5.18: Function Evaluations performed to find the global optimum for the $C_{identity}E_{cp}$ and $C_{pairwise}E_{cp}$ problem. All MBOAs fail. rHN-G and HC are successful.

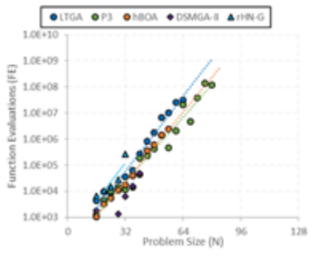
Figure Figure 5.19(a) and Figure Figure 5.19(b)

Figure Figure 5.20(a) and Figure Figure 5.20(b)

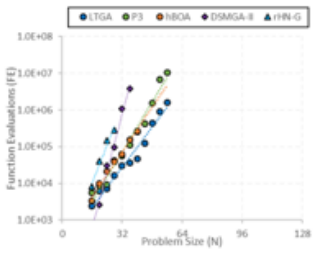
5.6.3 Model: Rescale Variations

In this section, the hierarchical environment function is used to evaluate the performance of an MBOA to recursively rescale the variation operator to higher orders of organization.

Figure Figure 5.22(a) and Figure Figure 5.22(b)



(a) Scalability performance of all SA-MBOAs and rHN-G for the $C_{lcoop}E_{cp}$ problem. All algorithms are successful in representing pairwise dependencies



(b) Scalability performance of all SA-MBOAs and rHN-G for the $C_{nlcoop}E_{cp}$ problem. All algorithms are successful in representing pairwise dependencies

FIGURE 5.19: Function Evaluations performed to find the global optimum for the $C_{lcoop}E_{cp}$ and $C_{nlcoop}E_{cp}$ problem. All MBOAs fail. rHN-G and HC are successful.

MBOA_Images/Experiment MBOA_Images/ExperimentalResults/Codd4_Ecp.pdf

(a) Scalability performance of all SA-MBOAs and rHN-G for the $C_{odd3}E_{cp}$ problem. All algorithms are successful in representing pairwise dependencies

(b) Scalability performance of all SA-MBOAs and rHN-G for the $C_{odd4}E_{cp}$ problem. All algorithms are successful in representing pairwise dependencies

FIGURE 5.20: Function Evaluations performed to find the global optimum for the $C_{odd3}E_{cp}$ and $C_{odd4}E_{cp}$ problem. All MBOAs fail. rHN-G and HC are successful.

Figure Figure **??** and Figure Figure **??**

5.6.3.1 Effect of Entanglement Depth

Figure 5.24(a) and Figure 5.24(b)

Figure 5.25(a) and Figure 5.25(b)

5.6.4 Discussion

An interesting observation from the results is that P3 and LTGA cannot exploit overlapping structure, yet for a shallow case, both algorithms where successful. Although the shallow definition creates a nonadditive separable problem, it appears that it is possible to approximate the structure using an additive separable decomposition. This is in line

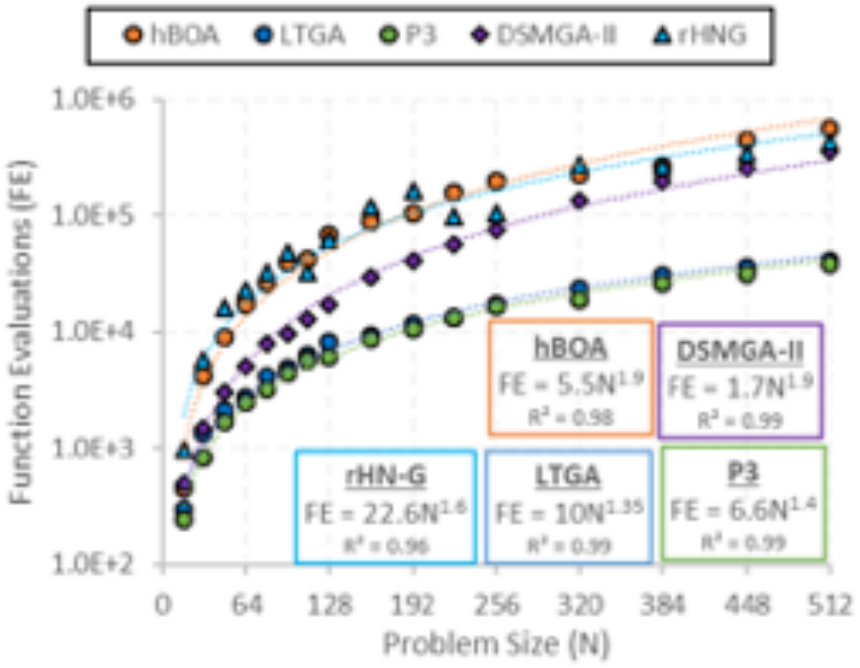


FIGURE 5.21: Scalability performance of all SA-MBOAs and rHN-G for the $C_{pair}E_{co}$ problem. All algorithms are successful in representing pairwise dependencies

MBOA_Images/ExperimentalResults/Clcoop_Eh.pdf MBOA_Images/ExperimentalResults/Cnlcoop_Eh.pdf

(a) Scalability performance of all SA-MBOAs and rHN-G for the $C_{lcoop}E_h$ problem. All algorithms are successful in representing pairwise dependencies

(b) Scalability performance of all SA-MBOAs and rHN-G for the $C_{nlcoop}E_h$ problem. All algorithms are successful in representing pairwise dependencies

FIGURE 5.22: Function Evaluations performed to find the global optimum for the $C_{lcoop}E_h$ and $C_{nlcoop}E_h$ problem. All MBOAs fail. rHN-G and HC are successful.

with results in the literature Pelikan et al. (2011); Goldman and Punch (2015); Hsu and Yu (2015); Chen et al. (2017). We discuss this further in Section ??. To improve our understanding of how a hierarchical problem increases the complexity such that it causes LTGA and P3 to fail, we perform an additional experiment that varies the depth of overlapping dependencies. These experiments use the hierarchical NSP definition and evaluate the performance of an MBOA with regards to changes in depth of overlap. The depth is controlled by shuffling subtrees within the hierarchical tree such that at depth d control variables do not have overlapping dependencies (detailed in Section ??). Figure ?? presents the results of the study. We see that for NSP_{hLO} LTGA and P3 are

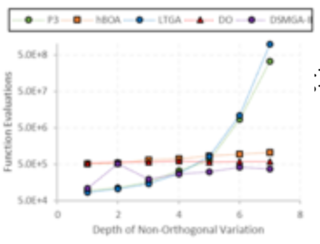
MBOA_Images/ExperimentalResults/Codd3_Eh.pdf MBOA_Images/ExperimentalResults/Codd4_Eh.pdf

(a) Scalability performance of all SA-MBOAs and rHN-G for the $C_{odd3}E_h$ problem. All algorithms are successful in representing pairwise dependencies

(b) Scalability performance of all SA-MBOAs and rHN-G for the $C_{odd4}E_h$ problem. All algorithms are successful in representing pairwise dependencies

FIGURE 5.23: Function Evaluations performed to find the global optimum for the $C_{odd3}E_h$ and $C_{odd4}E_h$ problem. All MBOAs fail. rHN-G and HC are successful.

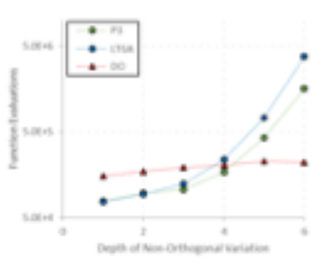
MBOA_Images/Experiment le_128.pdf

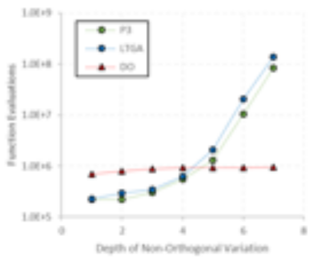


(a) Effect of entangled sub-functions for MBOAs in the $C_{lcoop}E_{hier}$ Problem size 12

(b) Effect of entangled sub-functions for MBOAs in the $C_{lcoop}E_{hier}$ Problem size 256

FIGURE 5.24: Function Evaluations performed to find the global optimum for the $C_{lcoop}E_{hier}$ for size 128 and 256. Depth of entangled sub-functions is varied.





(a) Effect of entangled sub-functions for MBOAs in the $C_{nlcoop}E_{hier}$ Problem size 128

(b) Effect of entangled sub-functions for MBOAs in the $C_{nlcoop}E_{hier}$ Problem size 256

FIGURE 5.25: Function Evaluations performed to find the global optimum for the $C_{nlcoop}E_{hier}$ for size 128 and 256. Depth of entangled sub-functions is varied.

sensitive to the depth of the problem, DSMGA-II, hBOA and DO are not. For NSP_{hNO}, we see a similar behaviour for LTGA and P3; they appear insensitive to the inclusion of nonlinear dependencies. hBOA however, is immediately sensitive to the inclusion of nonlinear dependencies, we were, therefore, unable to attain results for the problem size of 256 for hBOA. DO again remains insensitive. A critical element of the NSP construction is that it provides a fair comparison between different problem definitions. A change to the dependency structure in a subfunction does not affect the complexity of the fitness gradient. Equally, changing the complexity of the fitness gradient does not change the complexity of the dependency structure. The bivariate $\mathrm{NSP}_{(\mathrm{s/h})\mathrm{B}}$ and multivariate $\mathrm{NSP}_{(\mathrm{s/h})\mathrm{M}}$ cases provide a baseline for all MBOAs. $\mathrm{NSP}_{(\mathrm{s/h})\mathrm{B}}$ has similar complexity to previous benchmarks used in the literature Watson et al. (1998); Pelikan and Goldberg (2001). $\mathrm{NSP}_{(\mathrm{s/h})\mathrm{M}}$ is also included as a baseline for our analysis because cooperative overlap requires at least one of overlapping subfunctions to be multivariate. Therefore to attribute the change in performance specifically to cooperative overlap, we first must understand the performance of an MBOA on problems with non-overlapping multivariate subfunctions. All MBOAs are successful for both $\mathrm{NSP}_{(\mathrm{s/h})\mathrm{B}}$ $\mathrm{NSP}_{(\mathrm{s/h})\mathrm{M}}$. Most striking is the differences inefficiency. LTGA and P3 take a similar number of function evaluations and hBOA and DO take a similar number of function evaluations, yet the difference between these two sets is measurable. We hypothesis that this is due to the inductive bias of the models used. hBOA and DO use graphical models, whereas LTGA and P3 use a tree model. Therefore while hBOA and DO can represent a broader range of problem structures, constructing the correct structure can require more data. For LTGA and P3, their inductive bias is, therefore, stronger for exploiting a problem structure can be represented by a pairwise tree. Given that $\mathrm{NSP}_{(\mathrm{s/h}}\mathrm{B})$ can, then LTGA and P3 will be more efficient in exploiting this structure. DO and hBOA capable of representing the structure, however, the model space is larger and can, therefore, require more data to find the correct model for the problem structure.

The first key result differentiates the performance between pairwise models, LTGA and P3, from multivariate models, DSMGA-II, hBOA and DO. The differentiation is observed when the cooperative overlap is included: $\mathrm{NSP}_{(\mathrm{s/h})(\mathrm{LO/NO})}$. The difference between shallow and hierarchical fitness complexity highlights an interesting characteristic of P3 and LTGA. In the shallow problem definition, a method to follow the fitness gradient is to change one subfunction at a time. However, note there are no overlapping dependencies between pairs of subfunctions. Therefore the linkage tree can represent the pair of overlapping subfunctions as a single building-block. Given that P3 and LTGA use a model-informed variation, the algorithms are capable of finding the correct assignment for a building-block out of the possible four by the method of an exhaustive search. Therefore the scaling will be proportionate to 4^m where m is the number of pairs of overlapping subfunctions. This scaling does not cause failure. We see this less efficient approximation in the differences between the algorithms reducing when compared to the baseline, the exponent increases by 0.5, whereas hBOA and DO remain similar. (DSMGA-II)

When the problem challenge of rescaling the variation operator to a high-order of an organisation is introduced, the approximation used by P3 and LTGA is no longer sufficient. In the hierarchical fitness function, $\mathrm{NSP}_{\mathrm{h(LO,NO)}}$, the size of the variation doubles as the level of hierarchy to satisfy increases. Generally, as the linkage is random, as the size of the variation doubles, the number of overlapping subfunctions double. Therefore as the number of levels increase (doubling of the problem size) the size of the variation to make at the highest level of hierachy, and thus the number of subfunction not to disrupt, doubles. Specifically, the number of subfunctions that overlap with the control variables that undergo variation doubles. Therefore using the approximation method, we suspect P3 and LTGA use causes an exponential increase in the number of combinations of building-blocks that must be attempted in order to find the correct variation to make. Figure **??**.a ($\mathrm{NSP}_{\mathrm{(h(LO)}}$ supports this hypothesis. The number of levels of hierarchy is kept constant (7 levels of hierarchy in a problem size of 256), and the depth of the overlap is varied. Both P3 and LTGA show a sensitivity where the increase in the number of function evaluations performed to find the global optimum is close to double the change of measured at the previous depth. hBOA, DSMGA-II and DO however are insensitive to the change in depth.

The second key result differentiates the performance between using nonlinear hidden units or only visible units to represent relationships in the solution variables. DO is the only MBOA that uses nonlinear hidden units in the model. The first observation is that for the shallow fitness function, hBOA is unable to solve problems with nonlinear control variables, yet P3 and LTGA can. This result differentiates the ability to efficiently represent the partial solutions using the model and generate the novel solutions from the model. P3 and LTGA do not generate a solution or partial solution. Instead, they learn the linkage information only, the location of variables (the crossover mask) which interchange the values of variables between solutions. Therefore P3 and LTGA proceed as previously described irrespective if the pairs of subfunctions have linear of nonlinear dependencies. hBOA, on the other hand, must represent the dependencies in the Bayesian network as it aims to learn the joint probability distribution and exploits this information using model-informed generation. A Bayesian network is not efficient in modelling nonlinear dependencies as the number of dependencies grows exponentially with the number of nonlinear dependencies. hBOA is therefore not able to accurately represent the subfunction dependencies and is therefore unable to recombine partial solutions effectively failing. It is, therefore, no surprise that hBOA fails on the hierarchical fitness function. We have discussed that hBOA is indeed capable of dealing with overlap at all depths of the hierarchy. Therefore the cause of failure to hBOA is due to the presence of nonlinear dependencies. DSMGA-II (both shallow and hierarchical on NO)

Figure **??**.b and c provide further insight into the mechanism of failure due to nonlinear dependencies. P3 and LTGA fail due to the presence of overlap; the scaling between linear (Figure **??**.a) and nonlinear dependencies ((Figure **??**.b and c) is similar. When comparing the scaling behaviour of hBOA we see that in the nonlinear case (Figure **??**.b), hBOA show immediate sensitive to an increase in the depth of nonlinear dependencies. It is not the same behaviour observed for P3 and LTGA. Therefore the failure of hBOA is not a result of the same mechanism. For hBOA, it learns a generative model: the bayesian network captures both the linkage information and the value of the variable. Therefore the reason for failure is its inability to represent the dependencies of the subfunction accurately. Therefore in instances where the depth of nonlinear dependencies is low, an exhaustive search can find combinations subfunctions that combine to remove nonlinear dependencies. However, as the depth increases, the exhaustive search scales exponentially and therefore causes the failure to hBOA. Interestingly, when the depth of the problem is at its maximum (d=h), the number of function evaluations performed by hBOA, LTGA and P3 converge to a similar region. The difference between hBOA and both LTGA and P3 is what we would expect as an efficiency difference between the algorithms. To conclude, with regards to the state-of-the-art MBOAs we can attribute the failure of LTGA and P3 to overlapping dependencies only and hBOA to nonlinear dependencies only.

Finally, the results differentiate the performance of DO and all other MBOAs. It is not surprising that a deep neural network is capable of representing relationships between variables that other shallow models that do not use nonlinear hidden variables cannot. For bivariate and multivariate problems with no overlap do not requrie a deep network. However, for both $\mathrm{NSP}_{(\mathrm{h(LO)}}$ and $\mathrm{NSP}_{(\mathrm{h(NO)}}$ a deep network is required. Further experiments are performed that detial the relationship between using different depths of a deep neural network on the problem definitons explored in this paper in Appendix **??**. Importantly, and what differentiates DO from other attemps to use an autoencoder (shallow or deep) as the model in an MBOA, including a deep neural network has not caused results to be uncompetitive with other more straightforward problems in which some MBOAs are successful. As such, this is the first example of an MBOA using a deep neural network that is both competitive and outperforms state-of-the-art MBOAs. The results show that DO is the only algorithm that is successful in dealing with overlap and nonlinear dependencies.

A question that remains unanswered and of great interest to facilitate the practical use of MBOAs is the presence of characteristics explored in this paper to real-world optimisation problems. The problem characteristic of cooperative overlap is simple to construct: two functions that share the same arguments. We, therefore, must expect this to be present in natural/real-world optimisation problems. The definition of overlap has been previously defined in the literature; however, defining a problem that presents the expected problem challenges to an optimiser has been unsuccessful. NSP defines an overlap structure that creates these problem challenges; of course, other structures may exist that have overlap and create different, potentially more complex, problem challenges.

Table 5.1: The success or failure of an MBOA to find a global optimum solution in polynomial time. Deep Optimisation is the only method that is successful for all problem types.

Decomposition Type	Overlap Type	hBOA	DSMGA-II	LTGA	P3	DO
4*Shallow	None	✓	✓	✓	✓	✓
	LS	✓	?	✓	✓	✓
	NLS	✓	?	✓	✓	✓
	NLSNC		?	✓	✓	✓
4*Hierachical	None	✓	✓	✓	✓	✓
	LS	✓	✓	✓	✓	✓
	NLS	✓	✓			✓
	NLSNC					✓

The nonlinearly representable solutions are not a unique case. Given that the nonlinear subfunctions create partial solutions of one-hot assignments, we can expect to observe this challenge in assignment problems. However, the dependencies between modules explore in this paper can not be easily described using familiar optimisation languages. The problem characteristics explored in this paper are simplified and unrealistically well-behaved examples. A real-world optimisation problem is not going to contain only overlapping dependencies or only nonlinear dependencies. However, these results show what types of problem structure the models are and are not capable of exploiting. Given an optimisation problem that contains these characteristics, we can expect the algorithms to be unable to exploit this structure. However, the abundance of this structure will ultimately determine if that algorithm fails or not.

There is a considerable difference between the types of models, and therefore update methods and generation methods, used by the algorithms. Therefore the algorithms have a different cost of learning and generation associated with them. Deep-Optimization is the only algorithm that does not regenerate the complete model at each generation; it simply updates the existing model. Never the less, all algorithms have a polynomial time complexity for constructing and sampling the models. This paper focuses on differentiation the performance of what an algorithm can do (scaling polynomial) and what an algorithm cannot do (scaling exponentially). **Representation** Shallow cooperative overlap was insufficient to causes an exponential failure to LTGA and P3. This is because LTGA and P3 are capable of separating the building blocks and causing a variation between all solutions to the building-blocks. Therefore, it requires a random search to create the correct substitution to a building block. However, the correct substitution has a probability of 0.25 for all variations. Therefore, while less efficient, it does not require an exponential search. This inefficiency is represented in the difference between the pairwise complexity. Movement

Multi-level This is shown in chapter 5. DO can find the global optimum of hierarchical problems that have a strict tree nesting structure using a single layer network. – thus, a shallow representation of the multi-level structure. Whereas for problems that have a graph nesting structure (caused by cooperative overlap defined section x) require maintaining a separation of the low-level sub-functions until the global optimum is found, therefore requiring a multi-level representation. Further, problems that require exploiting the structure at all levels , where as for hierarchical problems that have greater than pairwise dependencies, a deep network is required. The model used by the algorithm must maintain the information of how past sub-solutions combined to form higher levels of organization to satisfy higher-order solutions (or sub-solutions)

5.6.5 Conclusion

It is important to know the problem structure; therefore, it is developed with clear structure. This allows for evaluating the performance of MBOAs. Understanding why it worked and or indeed failed.

This paper has aimed to differentiate the abilities of the different models used in state-of-the-art MBOAs. We have defined the NSP construction that facilitates precise control of problem complexity that causes a clear differentiation between the abilities of different models. The problem characteristic of cooperative overlap is introduced. Cooperative overlap causes both LTGA and P3 to fail. hBOA, DSMGA-II and DO are all successful, thus differentiating the capabilities between bivariate and multivariate models. As far as we are aware, this is the first result to do so such that pairwise models fail and multivariate model succeed. The problem characteristic of nonlinear dependencies causes failure to all state-of-the-art MBOAs. At the same time, DO is successful, thus differentiating the capabilities between using and not using nonlinear hidden units to represent relationships between variables. Further, DO is the only algorithm that is successful on all problem definitions explore in this paper. This result is the first to show that not only is a deep learning model used in a MBOA competitive with existing state-of-the-art MBOAs, but outperforms other MBOAs where all state-of-the-art MBOAs fail.

DO uses the Autoencoder model. Compared to the state-of-the-art architectures used in deep learning today, the Autoencoder is a relatively simple model. Nevertheless, with this model, DO has achieved state-of-the-art results. With the advancements in deep learning methods, by introducing a connection between MBOAs and deep learning via DO we provided an algorithmic framework that allows for exploration of other deep learning advancements to be applied to black-box optimisation. Of significant interest is the difference in deep learning architectures. It is accepted that different deep learning architectures are more suitable for learning tasks than others. We, therefore, expect the same for exploiting problem structure. Therefore, an important area of future research is with understanding how different deep learning architectures affect the inductive bias of an MBOA and whether it can exploit problem structure that other deep learning architectures cannot. Further, advance learning tools such as optimizers, regularizations, objective functions, inference methods that are widely accepted in the deep learning community can be applied to exploiting problem structure in optimisation problems. This is a new and open area of research.

To differentiate the abilities of the models used in MBOAs, we defined the Nonseparable Subfunction Problem (NSP). This construction enables explicit control of problem characteristics such that changes in a scalability analysis can be attributed to a change to the complexity of a problem characteristic. The NSP construction is an approach that allows for independent control of problem characteristics. It allows explicit exploration of problem characteristic complexities and facilitates a combination of problem characteristics. We have designed this construction to allow other problem characteristics, such as deception, to be included with relatively minor changes, if at all. This enables further exploration of a problem structure that can be explicitly and independtly controlled. However, as shown in the results, the construction detailed thus far was sufficient to differentiate the algorithms. We hypothesize that exploring more complex dependencies will create further challenges that only DO will provide an insight into the difficulty of the problem; other MBOAs are limited to the more straightforward cases of cooperative overlap.

Benchmarks used to evaluate the performance of MBOAs generally concentrate on a specific problem characteristic **???**. A short-fall of this approach is that we cannot evaluate an algorithms performance when a combination of these characteristics exists. It is through our research we identified that combining hierarchy and overlap into a single instance we are able to differentiate between state-of-the-art MBOAs. We call this theoretical benchmark problem the nested sub-functions problem (NSP).

overlapping building-blocks distinguishes the performance between current SOTA MBOAs and cyclic paths distinguishes the performance between EDAs and MSSA.

Chapter 6

Performance Evaluation of Deep Optimisation

6.1 Introduction

Chapter 4 provided the implementation details of DO. This chapter performs a performance analysis of DO. The objective of this chapter is to support the claims: 1. DO adapts the neighborhood of the solution space by re-scaling the variation operator and performs multi-scale MIV. 2. A deep neural network is required to represent problem structure that shallow neural network cannot represent 3. DO outperforms the state-of-the-art MBOAs in the follow three ways a. Following a non-linear solution trajectory b. Learning pairwise independent functions c. Exploiting entangled modes of variation to perform non-orthogonal variation

The chapter first starts by verifying the design decision made for DO. In addition, the chapter explores how DO is operating during the optimisation process supporting the claims the DO is performing multi-scale MIV. MBOAs are considered black-box methods. Therefore, understanding how they work is not straightforward. For DO, this is no different. Never-the-less what differs with DO from other MBOA is the availability of methods that can be applied to DO for providing an insight into how DO is working. Understanding how a deep neural network is computing the answer to the task is of great interest to the deep learning community. They are many tools developed to provide an understanding to how they are working. DO can use these tools to provide an empirical answer to 'what is the model learning' or 'what structure does this optimisation problem have'. Therefore, this chapter starts by exploring these tools usefulness into understanding how DO is performing. The theoretical problem developed in Chapter 3 has known structure allowing for empirically validation on the usefulness of these tools. The tools are taken directly from the machine learning community and are familiar in the literature to access the performance of new techniques introduced to the community citedenoising autos, contractive autos, regularized autoes etc. With this, DO enables evaluating the structure of unknown problems, as explored in Chapter 6. The performance of DO is analyzed using the theoretical problem developed in Chapter 3. The theoretical problem provides a range of problem characteristics to explore that are known to prove challenging to the state-of-the-art MBOAs. Further, the theoretical problem allows for clear and separate control of the problem challenges. Problems that are to be used for the results. - Dependencies complexity: Pairwise, Coop, 1hot, parity3 and parity 4 - Environment: Module, Hierarchical, Manifold

Further, we have demonstrated that using a deep autoencoder as the model is successful in problem structures that state of the art algorithms fails. We believe this is just the first step of connecting the ideas in deep learning to MBOAs and expect that other advancements made in this community can be beneficial to the evolutionary computing community.

6.2 Model Learning and Transitions

This section verifies model training and determining layer transition (As detailed in chapter 4). Determining when the induced representation is 'good enough' is not straightforward. However, this section demonstrates that this can be inferred by measuring statistics during training.

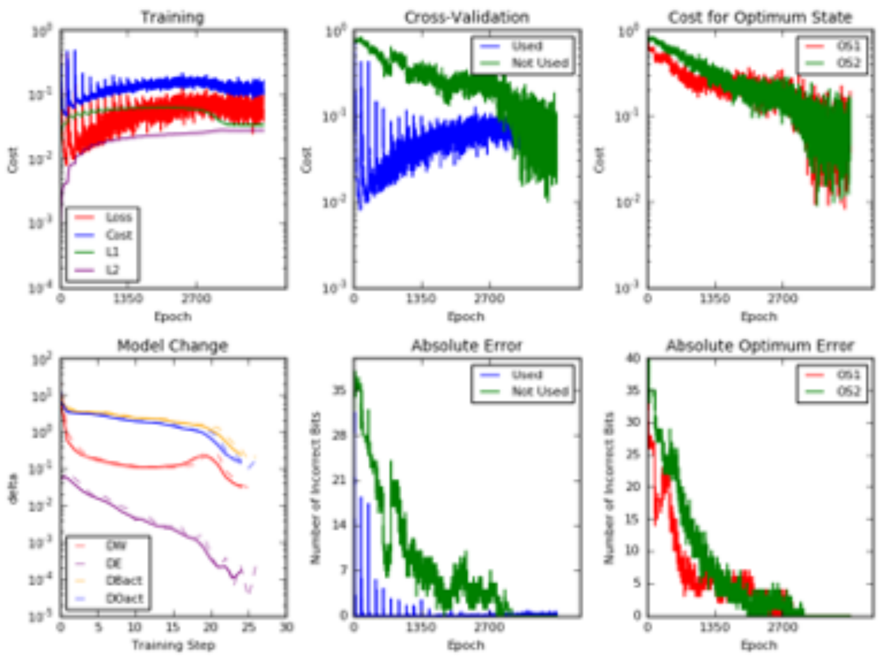


FIGURE 6.1: Example of the measurements taken during training used to verify the model performance.

Figure Figure 6.1 presents the measurements taken during training of the model. This figure is used throughout this section to compare model performance. The cross-validation measurement is performed by taking a sub-sample of solutions from the training set. These are not used during training. Each sub-graph presents the following measurements: Top left: The components associated with the training objective function. The cost is the objective value. The loss is the mean-squared error of the reconstruction. L1 is the L1 regularization cost. L2 is the L2 regularization cost. Top Middle: This graph compares the reconstruction cost for the training data (used) and the cross-validation data (not used). A model that has learnt a good general representation of the problem structure should have a low reconstruction error for unobserved solutions. Top Right: This graph presents the reconstruction error for the global solutions to the optimisation problem. A model that has learnt the structure that comprises of the global optimum should have a low reconstruction of the global optimum even if the global optimum has not been found. Bottom Left: This graph presents the change to the model. DW (delta weights) represents the relative change to the weights between two periods of training, a small change signifies a stable model. DE (delta error) represents the change to the reconstruction error before and after training the sample, a small change to the error signifies more data is not improving the model. DBact (delta Bottleneck activations) and DOact (delta Output activations) represent the change to the saturation of the activation response at the bottleneck and output respectively. Small changes imply the model is stable. Bottom Middle: This graph presents the absolute error for a solution from the training set (used) and cross-validation set (not used). The absolute error is calculated as the sum of the number of variables that are different between the input and binarized representation of the reconstruction. Bottom Right: The graphs represent the absolute error (as calculated for the bottom middle graph) for the global solution(s) for the optimisation problem. An epoch refers to the number of iterations per a training step. Unless otherwise stated, this referrers to the number of iterations of a complete training set. In Figure 1 however, Epochs represents the number training example used.

6.2.0.1 Online vs Batch Learning

Online training is used by rHN-G (predecessor of DO) to update the model. In ML a batch method is used to update the model. Here, the performance difference is measured between the methods.

The online learning method generates a solution and updates the model weights one solution at a time – a single solution is used only once in a backpropagation step. The batch method uses a definite training set-size and trained for a defined number training-set passes E. A single solution is used E times to update the model. Figure Figure 6.2 presents a comparison of the training performance between the online method and batch method for learning for the C_{pair} complexity of N32 (problem size 32). Epoch in the figure represents a single solution. The Batch learning case used a population size of 32 and 6 iterations of the training set. In the online learning case, almost 300 solutions are required to learn a representation that allows good reconstruction whereas the batch learning used only 32 solutions in total. This shows a significant advantage with regards to efficiency in exploiting information from the training data when using the batch method. To further confirm this, Figure Figure 6.3 presents a comparison of the scalability of the online learning method and batch learning method for the $C_{pair}E_{hier}$ problem. The performance improvement is greater than a magnitude improvement in function evaluations performed when using batch learning compared to online learning.

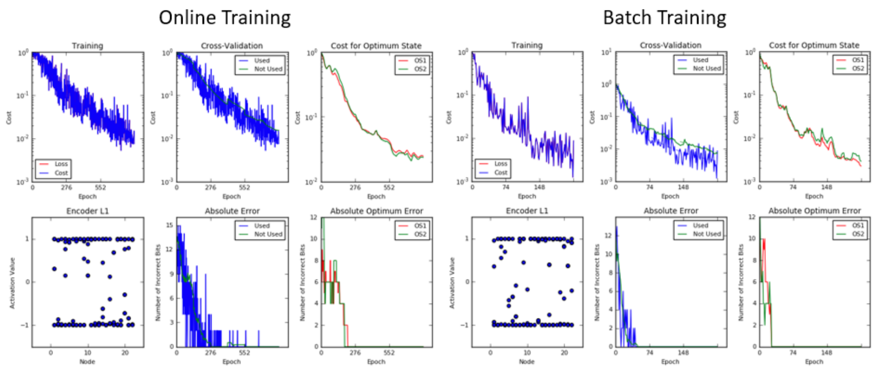


FIGURE 6.2: Learning performance comparison between online and batch for the $C = NO$ problem size 32

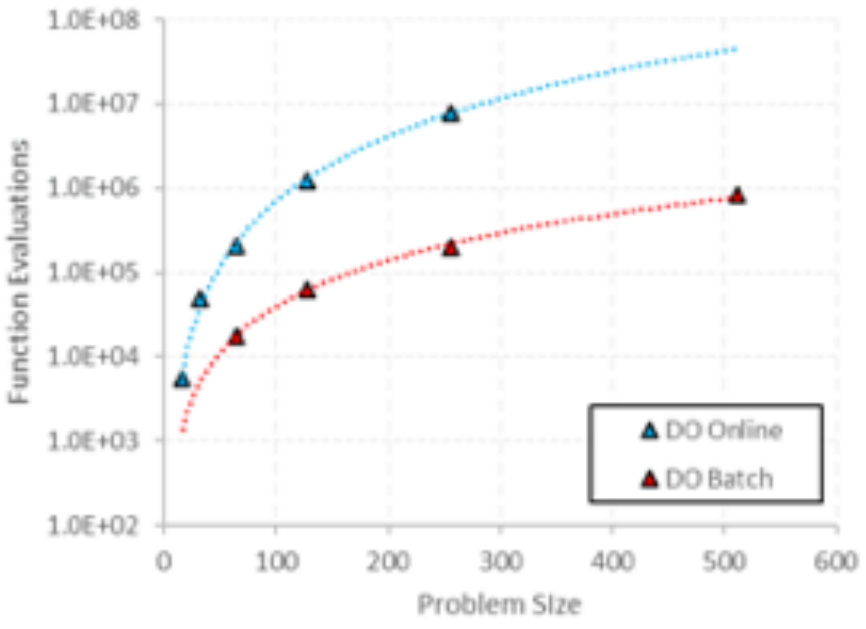


FIGURE 6.3: Scaling performance of Online DO vs Batch DO for the $E_{NO}E_h$ problem

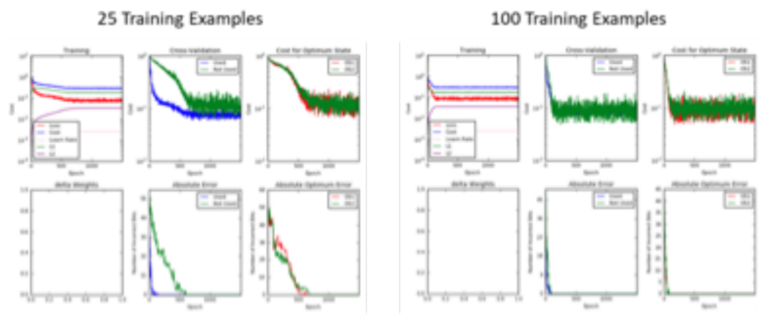
6.2.0.2 Distribution Size

Aim: Speed and practicality of using the algorithm. Developed to improve the speed for experimentation The number of data points that are required to learn a good representation of the relationships in the data is an important parameter. In MBOAs, this is set by the populations size. Determining the minimum population size in MBOAs requires running the algorithm until convergence and observing if an increase in the population size improves the best solution found. This can make the practicality of using such algorithms unfavorable as for large problems. As DO uses an autoencoder network, there are many tools available to measure the performance of the model and infer if more or less training data is required to improve the model. As such, manual tuning of the training data size can be performed without running the algorithm to completion. Further, it is possible to design empirical decision for determining a sufficient training set size during optimisation. As such, DO can run using automated transitions – the addition of more solutions to the training set is decided using empirical decision based on measurements taken during training.

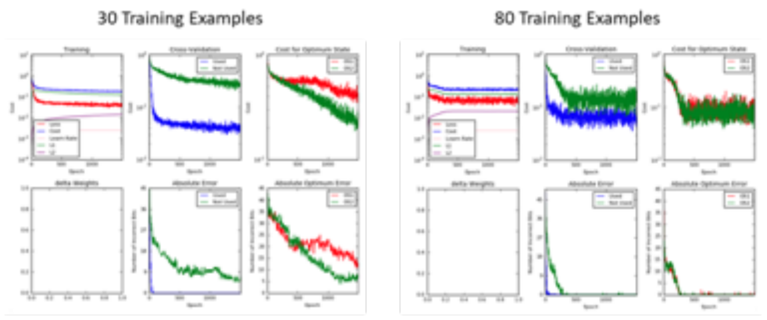
The size of the training set is sensitive parameter with regards to the performance of DO. Sufficient data is required to capture the relationships in the data distribution while the performance of and MBOA is measured by the number of functions evaluations performance, thus proportional to the number of solutions used. As Figure 2 demonstrates, performing many epochs of the same training set is efficient for learning a good representation. This introduces the additional parameters of number of epochs. Figure 4 and Figure 5 present the training performance for the Ppair 32 and Ppair 128 problem respectively. The effect of the training size and the number of epochs effects the learning time. It is sufficient to have a small training set size to learn the problem structure, however, the number of epochs required can significantly increase. While there are two hyperparameters that effect the training performance, the number of epochs is relatively insensitive. The number of training example on the other hand is critical to the success of training – increasing the number of epochs performed will not necessarily improve the model representation whereas increasing the training set size can.

Population Addition

As the number of epochs can remain constant, improving the automation of the training set size will significantly improve the practicality of DO. In this section, the usefulness of determine the size of the training set using the metrics that measure the performance of the model during training are evaluated. The method explored is incrementally increasing the size of the training set by adding additional training examples if the model has not learnt a good representation of the training set. This method introduces a new question, whether to update the existing model, or to reinitialize the model. This is evaluated by performing experiments that compare the training performance of the follow approach's: - Manual: Manually set training set size. - Reinitialize Model: 8 solutions



(a) A small training set size can lead to long training times. Comparison performed on C_{pair} size 128



(b) Effect of training data size. Validation errors and global errors show overfitting of a model during training when the data set is small. Comparison performed on C_{coop} size 128

FIGURE 6.4: Effect of training set size.

(batch size of 8) are added to the training set, the model is reinitialized, and all solutions in training set are used to train the model. - Update Model: 8 solutions (batch size of 8) are added to the training set and all solutions in training set are used to train the model. Experiments are performed using a problem size of 128 and relationships Ppair, Pcoop and P1hot. The manual setting of the training size if used as a baseline model. The two methods evaluated are the reinitialize model method and the update model method (detailed in Chapter 4).

Figure Figure **??**, Figure Figure 6.6 and Figure 8 present the results for the Pair, Pcoop and P1hot relationship complexities respectively. For all experiments it is observed that updating the model is more efficient than reinitialising the model when more training data to the training data set. For the complex relationship in 1hot, the difference is most notable where reinitialising the model never achieves the performance of the updated model. Compared to a fixed sized training set both Ppair and Pcoop require a similar cost. Further experiments would be required to further verify this cost as reported here is a marginal increase in the training set size required. However, for the 1hot complexity the update model is the only method to achieve a good generalisation of the training set. It is hypothesised that initially with a small training set puts the weights in a better basin of attraction even though the model has overfit. When new training examples appear with large errors associated with these examples (due to the overfit model) causes the

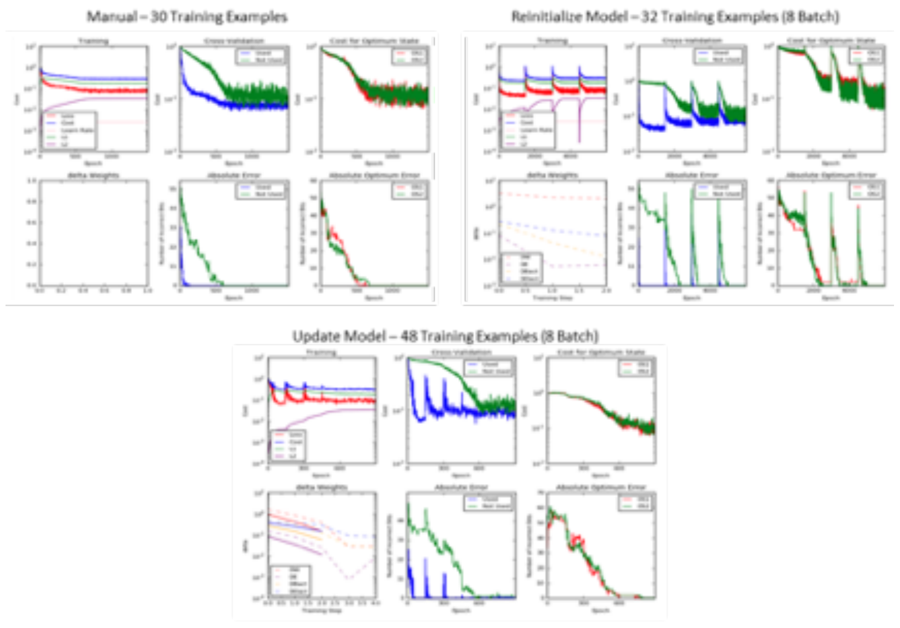


FIGURE 6.5: Comparison of learning performance for C_{pair} 128 when using manual training set size, batch addition with reinitialise model and batch addition updating model.

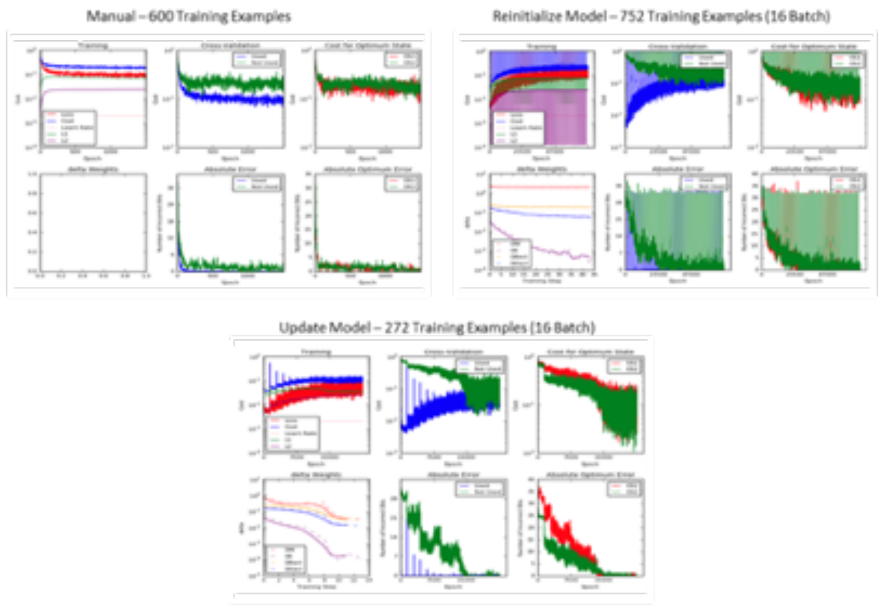


FIGURE 6.6: Comparison of learning performance for C_{nlcoop} 128 when using manual training set size, batch addition with reinitialise model and batch addition updating model.

weights to change. However, the change is starting from a better starting point then random initialisation, and thus allowing for finding a better model. Summary:

1. Model initialization can have a profound effect on the performance of learning.

Learning something about the relationships is likely to put the model in a better initialized weight space – seen in 1hot

2. Useful for deep models, as detailed later. Do not want to be re-initializing the entire model each time

3. Useful for automating transitions as we can now measure how the model reacted to new data. Did it cause it to adapt?

6.2.1 Automating Transitions

empirical - didnt really work, but some interesting progress to try improve the use of DO

We defined a transition when the layer, or rather model, is no longer adapting to improve the reconstruction of a solution.

It is found during the development of DO that an empirical transition criteria based soley on the cross-validation error was not robust to different types of problems. This is a common difficulty faced by machine learning practitioners as there are generally many local optimum in the cross-validation.

Measuring the performance of the model is relatively cost-free for a neural network model. How the measurements relate to a good performance is not as straight forward. Never the less, their exist multiple measurements in NN to allows assessment of the modes performance.

To improve the application of DO, an empirical method for determine the size of the training set has been developed. The method relies on adding a batch of solutions to the training set and measuring the change to the model performance caused by the additional solutions. As verified in the previous section, updating the model by adding a batch of training examples to the training set is suitable for learning a good representation. Aim: speed the process of using DO and applying to a wide range of problems. Can then be manually tuned after initial exploration using automation. Empirically developed to set the number of training examples required for the model to learn a good representation of the model. The methods for deciding transition are detailed in Chapter 4. The experiments uses the automatic transition criteria to decide when no more training data will improve the relationships learnt by the model. All experiments Figure 9 presents the performance during training for problem complexities $C = NO$, $C = LO$ and P1hot for sizes 32, 128 and 512. The empirical decision based on the changes to the model provided a conservative transition decision – no run decided to transition before a good representation was reached while in some cases an earlier transition could have been performed.

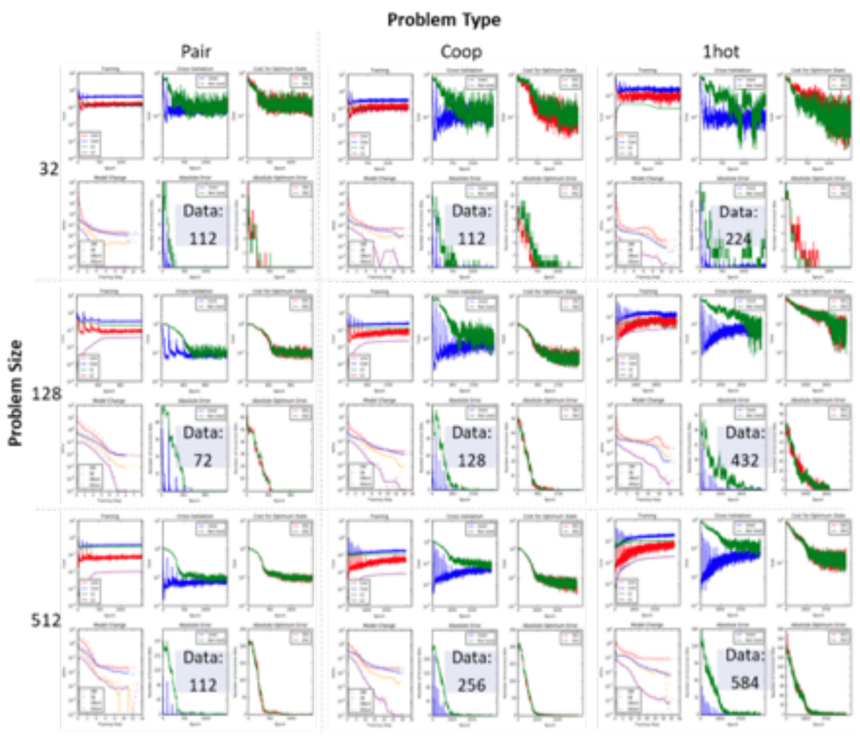


FIGURE 6.7: Performance of automating transitions based on empirical measurements shows a conservative transition.

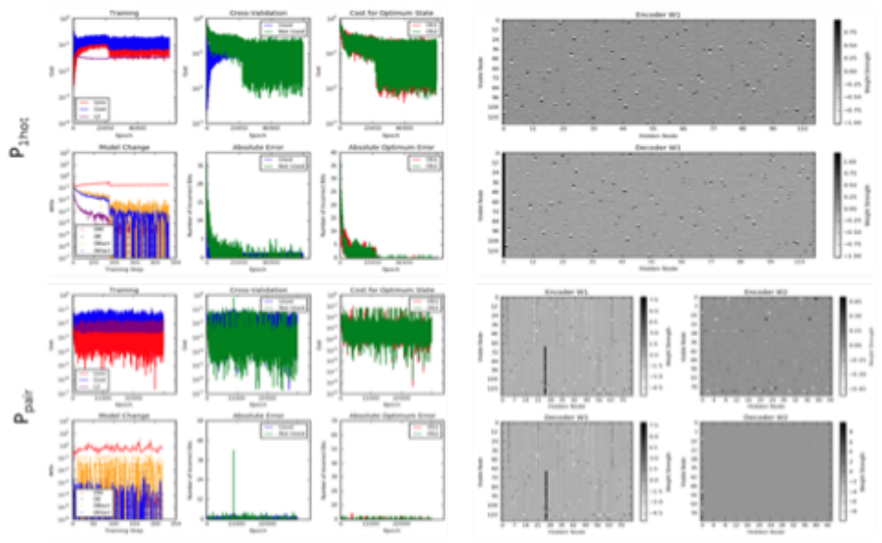


FIGURE 6.8: Example of failed transitions that indicates that the hyperparameters used are not suitable.

Figure 6.8 demonstrates failure of the transition method. Here, the criteria was not met to make an autonomous decision of transition. Generally, this occurs due to oscillatory behaviour in the model. Empirically, this is observed when the regularisation pressure is to large, causing the weights to be small, or the learning rate is to high. While a failure, it provides important feedback for hyperparameter tuning. As generally performed in machine learning, determining whether the model has been sufficiently trained is achieved empirically. The transition method developed for DO is empirical and therefore is not as efficient as a manually tuned training set size. Nevertheless, it is conservative with the decision such that early transition is avoided. Most note-worthy is that DO allow for multiple measures of the model performance during optimisation. This is missing from the SA-MBOAs. The algorithm is run to completion to determine if the population size is adequate. In DO, this is not necessary as one can measure the model's performance during the algorithm run. Further, empirical measurements of the model performance have been developed that allow for monitoring of the performance of automatically determining the transition time for the model.

6.3 Representation Learning

DO is unique compared to other SOTA-MBOAs as the parsimony pressure applied can be controlled by hyperparameters. In this section, the effect of pressure applied the model for DO is evaluated. This provides an empirical understanding for how the parsimony pressure effects the representation learnt for DO and what type improves the performance of DO.

The performance of DO is expected to be sensitive to the representation of the latent space. Ensuring a good representation will provide a more efficient information exploitation and therefore solution exploration. In this section, what makes a good representation is first explored. Then, an empirical investigation explores how the parameters available to DO during training affect the induced representation. To understand the performance of DO to induce a representation the model is evaluated by accessing how the model represents the relationships for the theoretical problem complexities. As discussed in Chapter 4, regularization methods can have a significant effect on the representation learnt be the autoencoder model. Therefore, this section investigates the effect on the induced representation. To evaluate the performance of the model, the following measurements are taken for all problem complexities defined in chapter 3:

1. Weights: The weights of the neural network provide a measure for the relationships learnt between the variables. The weights are often used to illustrate the 'filters' learnt by a model cite(denoising ae, contractive). As the theoretical problem has known problem structure the learnt weights provide a measure for how accurate the structure has been captured and how the model is capturing the relationships.
2. Accuracy of the representation: Learning a compression of the directions of variation effectively apply a restriction on the degrees of freedom for variation in order to allow for small variations to travel larger distances. An accurate representation is one that does not restrict the degrees of freedom for generating sub-solutions to the sub-functions.

3. The adapted solution neighborhood: The latent representation adapts the neighborhood space for a solution. Accessing the solution neighbors in latent space provide an insight to the accuracy of the relationship learnt. Neighboring points in latent space should relate to meaningful changes in the solution space.

6.3.1 Building-block

In this case, two neurons must work together to represent an accurate dimensional compression of the solution space. Unlike the pairwise only case where a single neuron is sufficient to represent a compression and separation of independent building blocks.

How the model can represent the relationships of a sub-function is first explored. The problem size is set to 4 (a single sub-function). The autoencoder model is set the minimum capacity required to compress the input space i.e. 4 solutions to be represented by 2 hidden units. The runs are performed using training examples that are optimal for a sub-function. The number of training examples used is 1000 (enough for learning all compression). No denoising or regularization is not used for these experiments.

Figure 11 presents the results for sub-functions pairwise, cooperative overlap, one-hot and odd-parity4. All complexity functions were representable by a single hidden layer. For each complexity function there are six graphs that verify the suitability of the representation. Error plot: The top left graph presents the reconstruction error for each sub-function solution and the average reconstruction error (the training objective). All solutions have a low reconstruction error. Activation Response: The bottom left graph presents the activation response for each sub-function solution. The latent response has saturated activation values where each encoding of a solution is distinct for all others.

Weights Plot: The top middle graph presents the weights for layer 1. This provides information on the type of relationship learnt or the filter applied by a neuron. This filter mapping will be used to evaluate the performance of larger problem size as one should expect to observe weights with multiples of these relationships maps. Generation Plot: The bottom middle graph presents the reconstruction output (actual decoder unit values) for all combinations of the hidden representation with values limited to the minimum and maximum activation response. E.g. for pairwise this is (11), (1-1), (-1,1) and (-1,-1). This shows that each solution to the sub-function can be generated by using a discrete latent representations.

Interpolation Plot: The top right graph presents the interpreted decoding output of a linear interpolation between four latent representations. At index 1 the activation value is 11 at index 2 the activation value is -11, at index 3 the activation value is -1-1 and at index 4 the activation value is -11. 10 linear interpolation steps are taken in-between each index (at the latent representation). This indicates the behavior of moving along a single activation values response. This plot demonstrates the neighborhood between each sub-function solution. An ever spread signifies a good separation of the variation.

Generation Interpretation Plot: The bottom right graph presents the same as the bottom middle graph but with the decoded outputs interpreted using a threshold of the middle output activation value. This threshold method is used by DO for MIV. Thus this plot validates the assumption of using the threshold interpretation.

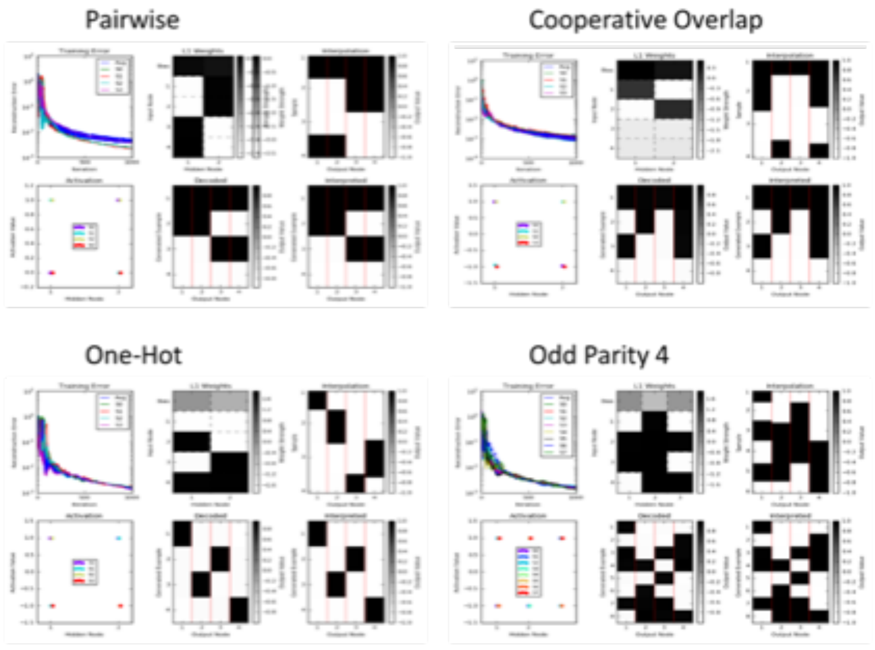


FIGURE 6.9: Response for learning a representation of the relationships in a single sub-function defined in the theoretical problem for complexities pairwise, cooperative overlap, one-hot and odd parity 4.

Figure 6.9 shows that all sub-function solutions can be compressed using a single hidden layer. Note for parity 4 there are 8 solution to a sub-function.

Figure 6.10 represents the representation learnt for the parity 3 function. The parity 3 complexity not only has a pairwise independence complexity, but it also requires a two-layer network to compress the representation. Figure 6.10.a uses a single hidden layer network with two hidden neurons. The model is only able to capture three of the 4 solutions to the sub-functions as seen in the Error plot, Generation plot, interpolation plot and generation interpretation plot. Figure 6.10.b is the results from a single-layered model using 3 hidden units. The model is only capable of learning the identity function and the pairwise correlation between variables 3 and 4. While the space has been compressed from 4 DOF to 3 and accurately represents the solutions to the sub-functions it has failed to improve the neighbourhood space. This is observed in the interpolation plot. No two solutions to the sub-function are neighbours in the latent space, to reach an alternative sub-function solution requires an intermediate step via an assignment that is not a solution to the sub-function. Therefore, local movements in latent space will not be

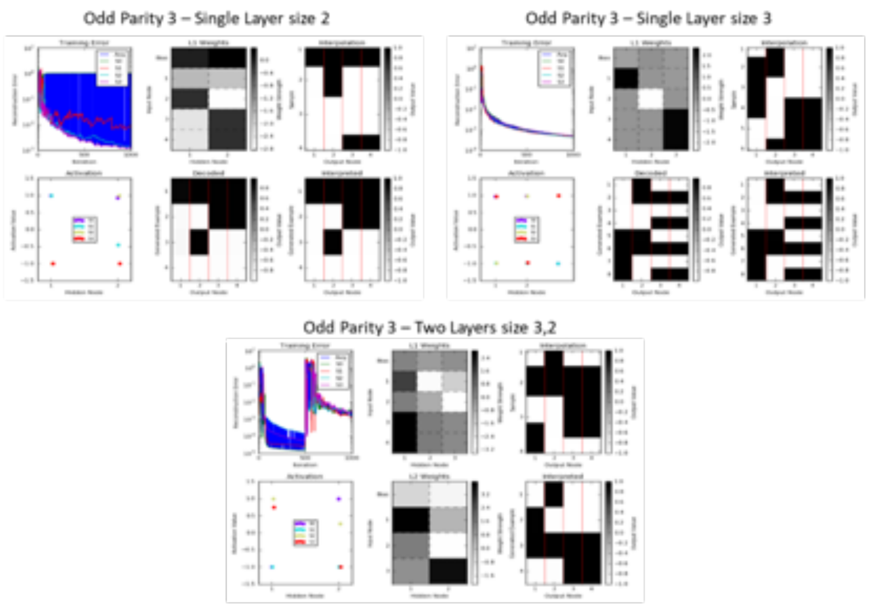


FIGURE 6.10: Response for learning a representation of the relationships in a single sub-function defined in the theoretical problem for complexities odd parity 3

enough to make the required changes in the solution space. Figure 6.10.c uses a two-layer model with 3 hidden nodes in the first layer and 2 hidden nodes in the second layer. The model is training using the layer wise method as visible in the error plot. The model can compress the solution space to a dimension of 2. The reconstruction plot is replaced by a plot of the weights for the second hidden layer. The interpolation plot shows that sub-function solutions are now neighbours in latent space. The Generation Interpretation Plot verifies that each sub-function solution is generatable using a combination of the activation values extremes.

6.3.2 Learning a Representation

Denoising and regularization are recognized techniques used by machine learning practitioner to improve the performance of the model. For DO, the metric is the representation learnt. In this section, the effect of dropout, drop-connect, l1 and l2 regularization are empirically evaluated for each problem complexity. In these experiments a problem size of 32 is used throughout. It is important to not only represent the relationships for a sub-function but to also separate one sub-function form another. The experiments are performed on a range of models using the theoretical optimisation problem of size 32 (8 sub-function modules). The aim to is understand the importance of dropout, drop-connect, l1 and l2 regularization for inducing a representation that is efficient for MIV. A grid search is performed for each problem complexity and for each combination of denoising and regularization. The parameters for drop-out and drop-connect rates are consistent throughout all experiments area set to 0.8. The L1 and L2 regularizations were manually tuned to allow for most efficient representation for each complexity class. The evaluation is made by observing the weights matrix and individual neuron vectors and observe how similar or dissimilar they are to the representations presented in the previous section. Additionally, the available of each sub-function solution is measured for each sub-function by randomly sampling the latent space, decoding the representation to for a reconstructed solution and measuring the frequency of solutions for each sub-function. To provide a clearer differentiation between the performance, the training set size is fixed for withing a complexity class. The training set consist of local optimum solutions as found by using the random-restart hill-climbing algorithm. This is achieved by restricting the variation to the original solution space and manually tuning the number of training examples available in the training set. The representation performance for deep models required for disentangling deep sub-functions is accessed separately in section (DO Performance). The experiments performed in this section is investigating the ability to learn the relationships defined by the complexity function only.

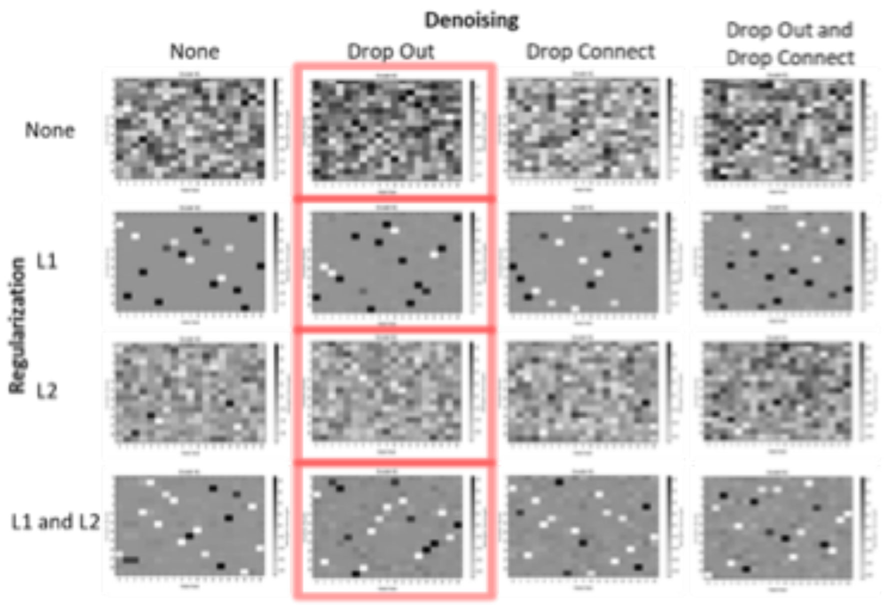


FIGURE 6.11: Comparison of Layer 1 weights learnt for the for $C = NO$ Complexity problem size 32 (8 sub-functions).

Figure 6.12 presents the weights for the grid search using a total of 32 training examples. The L1 regularisation provides a clear separation of variation directions as seen in the L1 regularisation row. The L2 regularisation does not provide any significant improvement compared to the representation without any regularisation. Dropout and Drop-connect do not have any significant effect on the representation (seen in all denoising columns). The representation is further verified by measuring the available solutions and interpolating between the global optimum solutions as presented in Figure 14.The figures are generated using the models highlighted in red in Figure 6.12. The available solutions plot is generated by randomly sampling the latent space a total of 30 times and calculating the frequency of each solution to a sub-function. The bar chart is separated into each sub-function (building block). The blue colour represents an invalid solution to the sub-function. The purple, red, green and light blue represent each of the 4 solutions to the sub-function. The none-hashed bar represents solutions randomly generated using the solution level representation. The single hashed bar represents solutions form the 1st hidden layer. It is observed that at the solution level each individual solution is represented evenly and take up around 25% of the frequency (as expected). Generating from the 1st hidden layer, there is a considerable change to the generatable solutions. For each regularisation, almost all solutions generated have valid solution to the sub-functions. Further, using the L1 and L2 regularisation in combination provides the most even spread. Thus, indicating that L2 can help improve the access to sub-function solution.

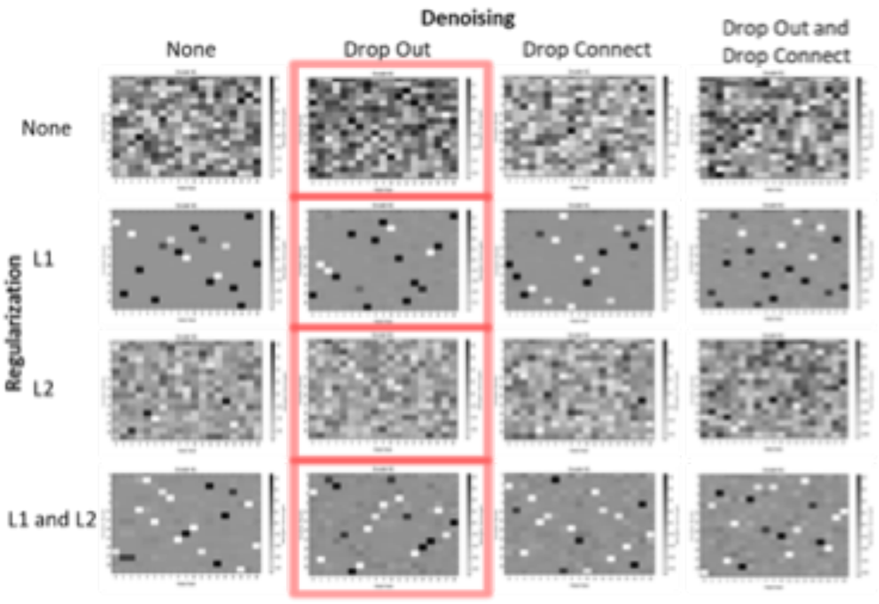


FIGURE 6.12: Comparison of Layer 1 weights learnt for the for $C = NO$ Complexity problem size 32 (8 sub-functions).

The interpolation plot for each model is presented in Figure 14. The interpolation plot is generated by taking one global optimum solution and for each unit replacing it with the unit of another global optimum solution. The interpolation is performed with representation at the solution level and the first hidden level. The plots show the solution generated and the fitness of that solution after each swap. For the Drop-Out only cases, the interpolation performed at the first hidden layer is noisy – a sub-function solution is repeatedly swapped. Further, the fitness of a solution is also noisy – there is a not a clean interpolation between the two global optimum solutions. This suggest that solutions will still get easily trapped when searching in the space defined by the first hidden layer. In the DO and L1 case the interpolation is clean, a variable at the solution level is only changed once during the interpolation and the fitness of the solutions in between have a smoothed fitness - the fitness landscape resembles the environment function.

In summary, using the L1 regularisation is most beneficial for the pairwise complexity and showed no apparent sensitivity to the denoising methods used.

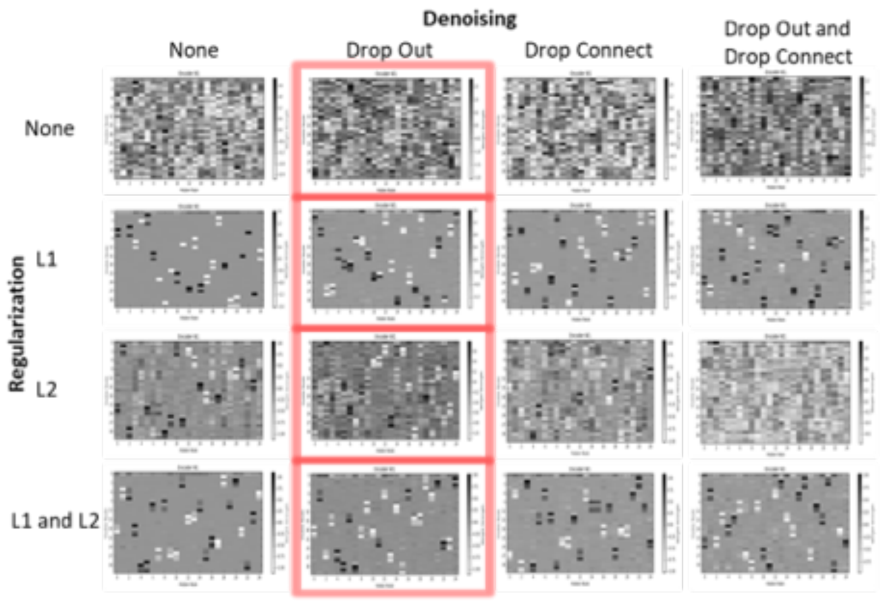


FIGURE 6.13: Comparison of Layer 1 weights learnt for the for $C = LO$ Complexity problem size 32 (8 sub-functions).

Figure 6.13 presents the model weights when training using local optimum solutions for the Pcoop problem size 32. 80 training examples are used for each experiment. The observations made are:

- L1 provides a clear separation of the variation directions. However, without denoising, some hidden units learnt the identity.
- DO and or DC improve the representation as they encourage not to learn the identity (comparison clear in L1 row), also visible in L2 and L1 and L2 rows.
- L2 provides a good separation of building blocks, not as clear as L1, however, allows for accurate representation of the sub-function solutions as it does not favor neurons with the identity function. Dropout appears to improve the representation compared to only L2 regularization

Figure 6.14 presents the plot of the generatable solutions for each building-block and the interpolation between two global optima. The plots correspond the to the models highlighted in red in Figure 15. The observations made are:

- DO only does not provide an accurate representation, for building-block 8 solution number 2 was not generated. Generation in other blocks is also not well distributed, therefore it is likely that variations to the modules answers will be highly unlikely while other favored.

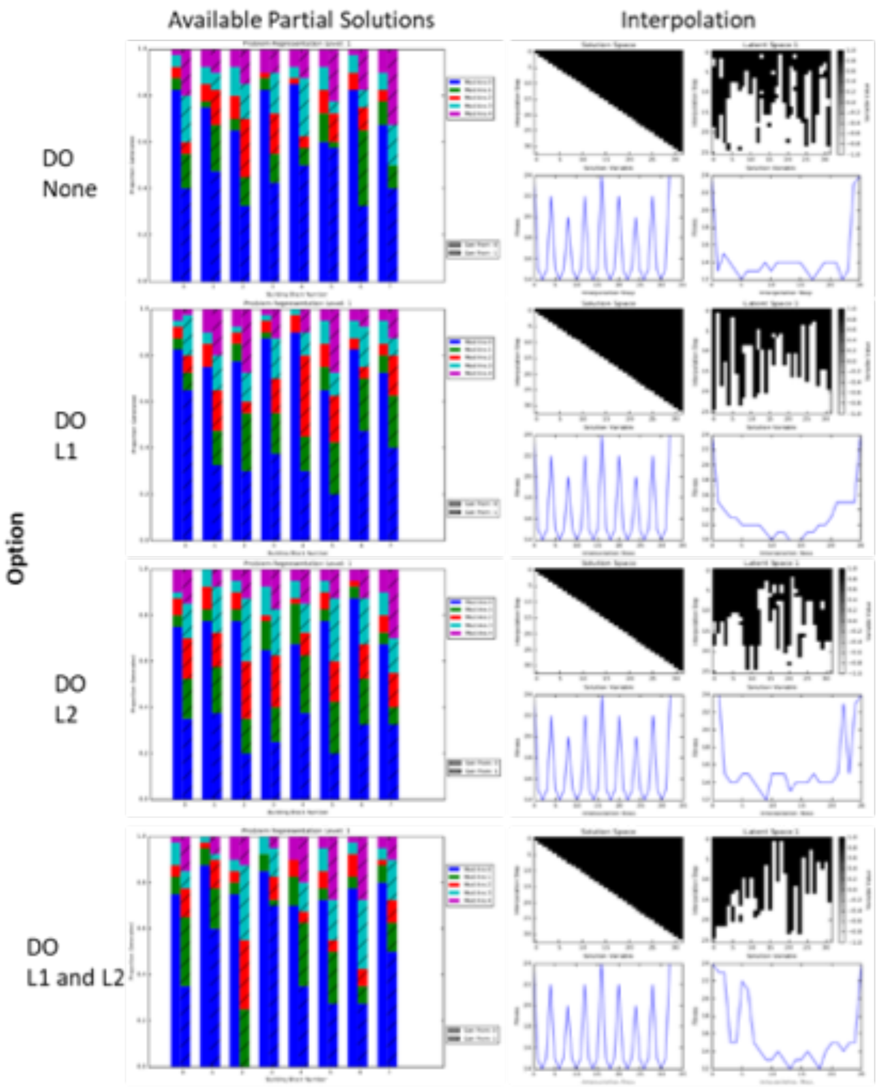


Figure 6.14: Solutions generatable by DO and interpolation between the two global optima for denoising option dropout.

- L1 also does not provide an even distribution while L2 provides an good even distribution of generating each solution for each sub-function.
- L2 alone does not provide a clean interpolation, therefore search in the latent space is likely to get easily trapped when using MIV. L1 on the other hand provides a clean interpolation.

In summary, L1 and L2 provides a good balance between separation of the building-blocks while accurately representing the sub-function solutions. Dropout and or drop connect show improvement over no denoising.

Using 160 Training Examples

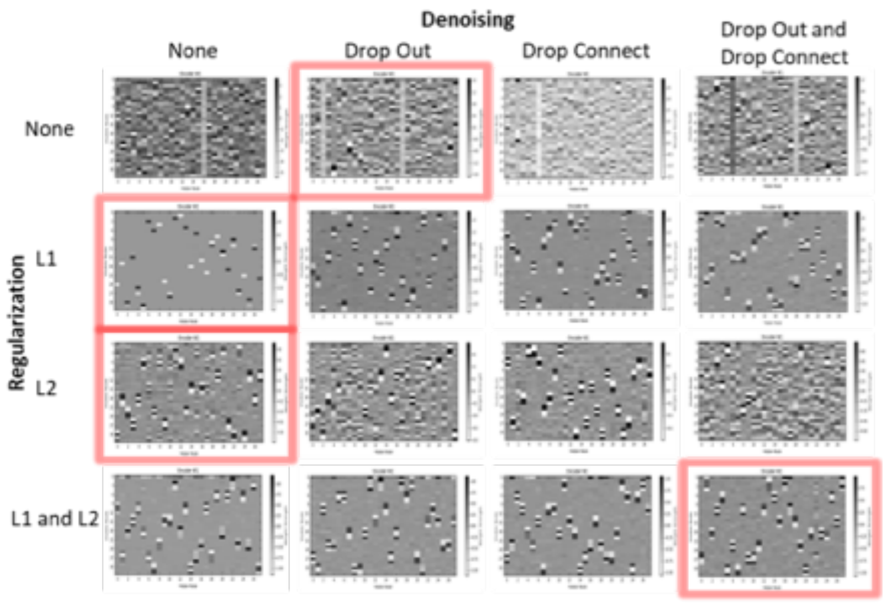


FIGURE 6.15: Comparison of Layer 1 weights learnt for the for $C = NLO$ Complexity problem size 32 (8 sub-functions).

Figure 6.15 presents the model weights when training using local optimum solutions for the P1hot problem size 32. 160 training examples are used for each experiment. The observations made are:

- DO and DC have a clear positive effect for this problem complexity.
- For this problem complexity, the difference between L1 and L2 regularization are amplified in comparison to $C = NO$ and $C = LO$. L1 favors sparsity while L2 favors a spread of the connection strengths. For P1hot, multiple hidden units must have 4 connections of similar weights to accurately compress the space. As such L2 is a more suitable regularization. This behavior is observed in the weights where L2 performs better that L1 for learning a compressed representation.
- The L1 regularization improves the separation of building-blocks but without DO or DC the model learns a representation close to the identity function.
- DC improves the representation when compared to DO only. This is because DC favors multiple connections in the representation which is favorable for the P1hot problem. It causes response to the representation similar to that of L2 regularization.

Figure 6.16 presents the plot of the generatable solutions for each building-block and the interpolation between two global optima. The plots correspond the to the models highlighted in red in Figure 6.15. The observations made are:

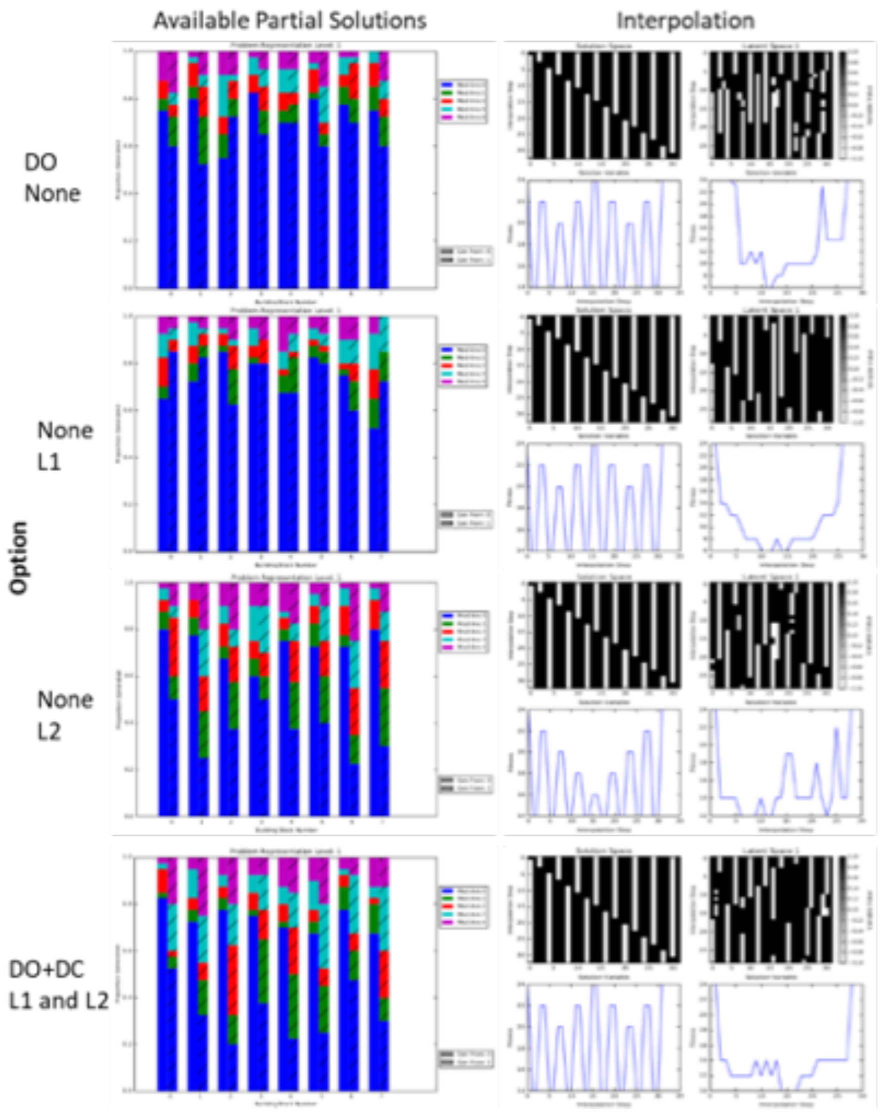


FIGURE 6.16: Solutions generatable by DO and interpolation between the two global optima.

- The use of L1 only learns a function similar to the identity function, as such, generating solutions from the latent space provides no improvement in comparison to generating at the original solution level.
- L2 and DO+DC+L1+L2 have a similar performance with regards to generatable solutions. DO+DC+L1+L2 however has a cleaner interpolation.

In summary, L1 is only useful when used in conjunction with DO, DC or L2 as it helps to separate the building-blocks. There is a similar performance between DO, DC and L2

Figure 6.17: Comparison of Layer 1 weights learnt for the for $C = Odd4$ Complexity problem size 32 (8 sub-functions). Figure 19 presents the model weights when training

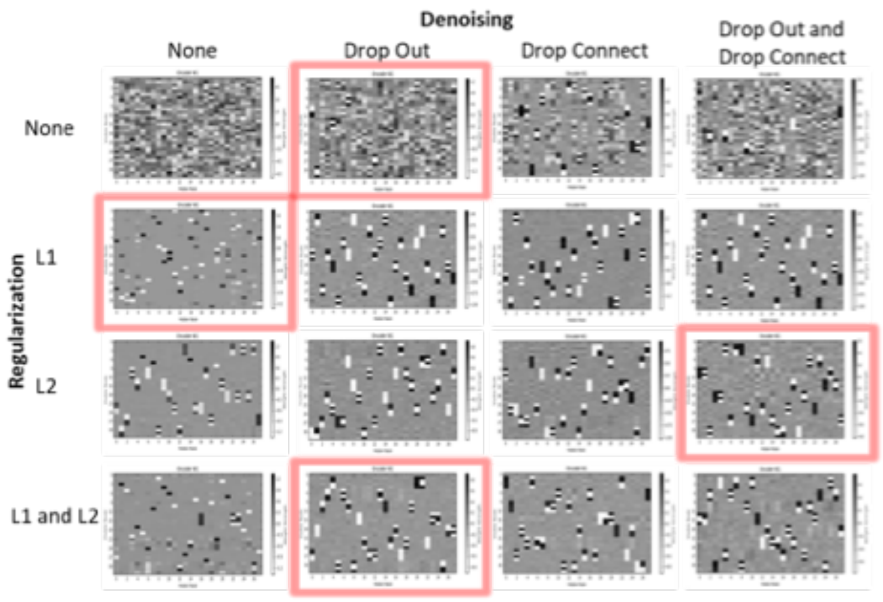


FIGURE 6.17: Comparison of Layer 1 weights learnt for the for $C = Odd4$ Complexity problem size 32 (8 sub-functions).

using local optimum solutions for the P1hot problem size 32. 256 training examples are used for each experiment. The observations made are:

- Similar to the 1hot case, the L1 regularization causes the model to learn the identity function. Whereas when used in conjunction with DO and or DC the representation is good.
- L2 performs well at separating the sub-functions and performs best when used in conjunction with DO and or DC.
- DO and DC have a significant impact on providing a good representation.

Figure 6.18 presents the plot of the generatable solutions for each building-block and the interpolation between two global optima. The plots correspond the to the models highlighted in red in Figure 17. Note that there are 8 valid solution available for the $C = Odd4$ function, however, to keep the graph visually interpretable a solution and its complement are grouped together in measuring which solutions to a sub-function are available. The observations made are:

- Generating solutions using the L1 regularization only model performs similar to generating solutions from the original solution representation and therefore has not compressed the search space. Interpolation is noisy and therefore the neighborhood defined at the latent representation is poor for searching via local variations.
- The is a negligible difference between DO+DC+L2 and DO+L1+L2 when comparing the generatable solutions plot. Comparison of the interpolation plot reveals

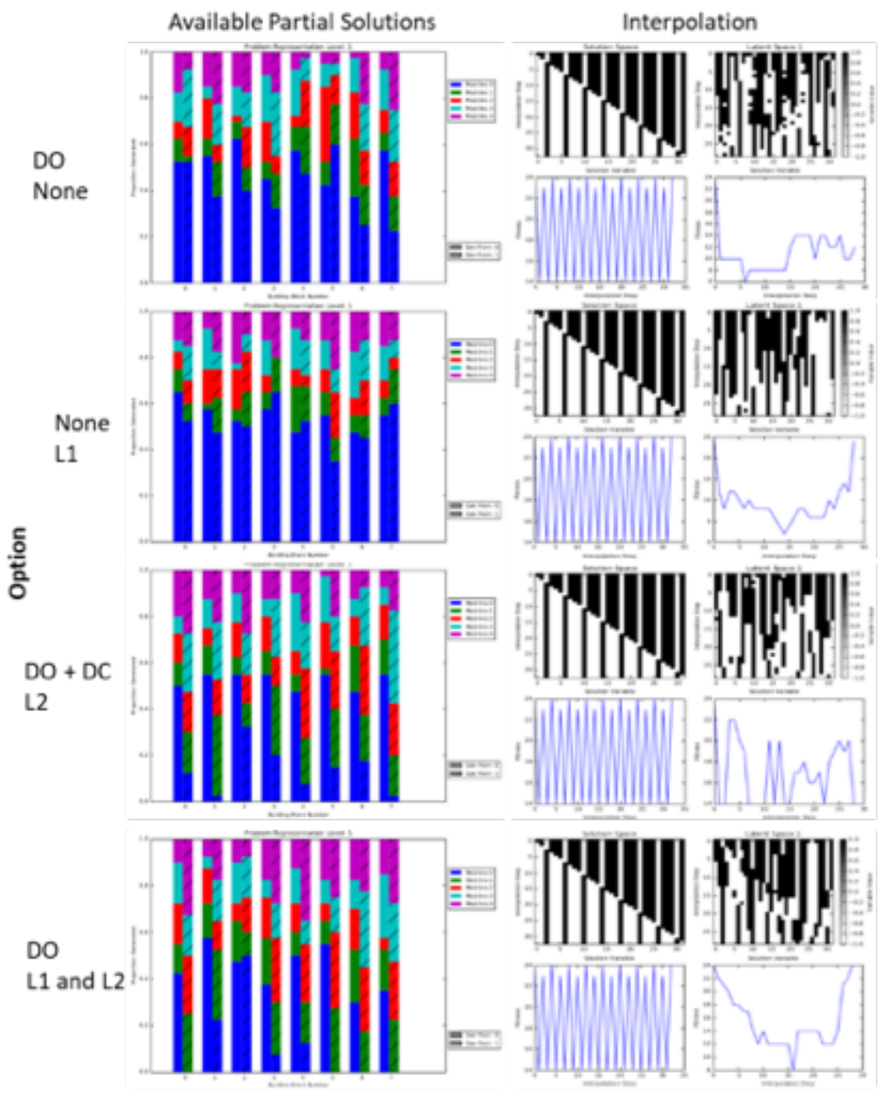


FIGURE 6.18: Solutions generatable by DO and interpolation between the two global optima.

that DO+L1+L2 has learnt a better neighborhood as the noise in the interpolation with regards to both the solutions generated and their corresponding fitness.

In summary, L1 is only useful when used in conjunction with DO, DC or L2 as it helps to separate the building-blocks. There is a similar performance between DO, DC and L2.

As presented in section [sub-functions] the complexity Podd3 requires two hidden layers to compress the representation space. For this experiment two hidden layers are used in the model. The model is training using the layer wise technique using local optimum solutions for the Podd3 problem size 32. 320 training examples are used for each experiment. Figure 21 presents the weights for layer one after training only the first layer. Figure 22 presents the weights for both layers after training both hidden layers where

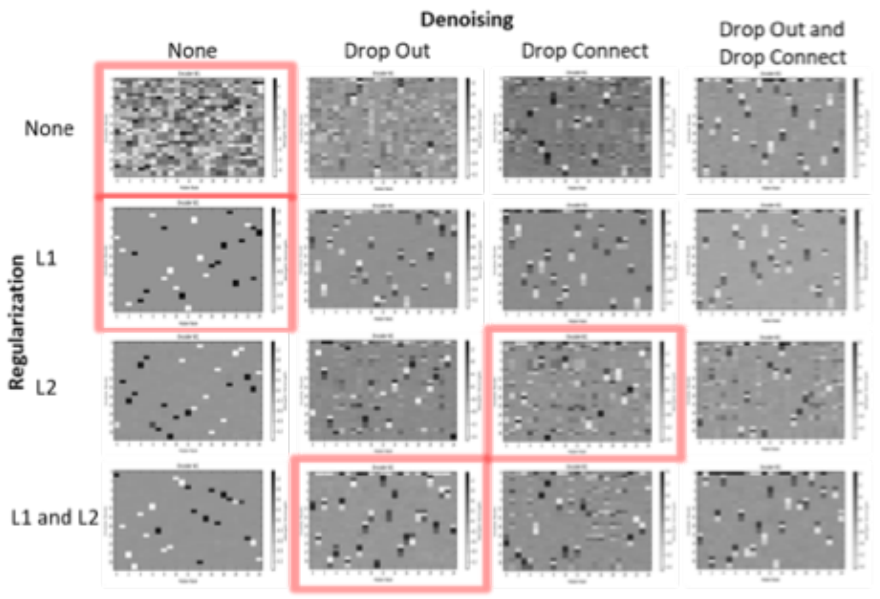


FIGURE 6.19: Comparison of Layer 1 weights learnt for the for $C = Odd3$ Complexity problem size 32 (8 sub-functions).

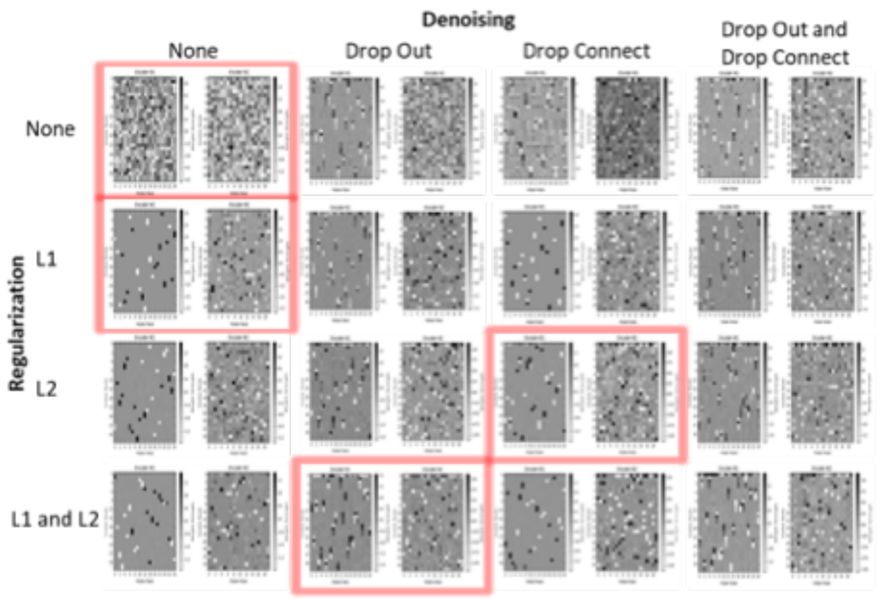
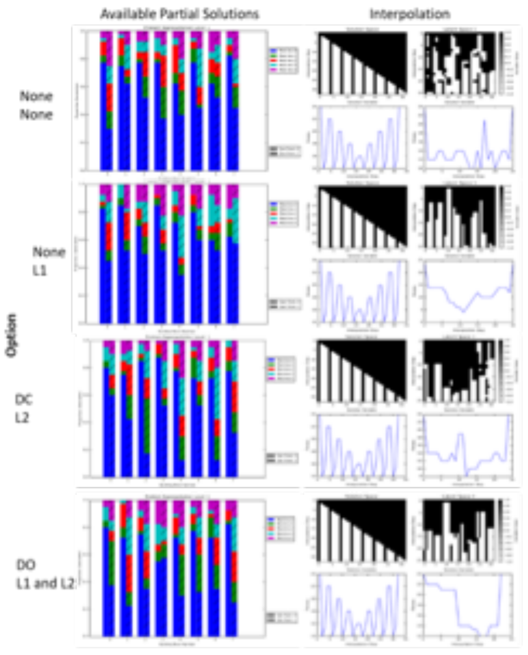


FIGURE 6.20: Comparison of Layer 1 and Layer 2 weights learnt for the for $C = Odd3$ Complexity problem size 32 (8 sub-functions).

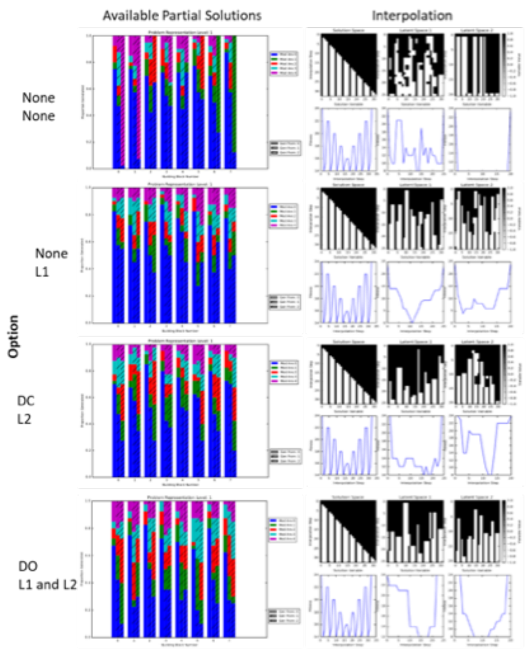
the first hidden layer is initialised with the weights found from training the first hidden layer only. The observations made are:

- Both the L1 and L2 regularisation without DO or DC are unable to learn a good representation of the solution space.
- DO has a very positive effect on the performance of the representation, without regularisation it performs well. With regularisation the representation is enhanced.

- DC does not perform well especially when combined with either regularisation.
- Using DC along with DO provides no significant advantage.



(a) Solutions generatable by DO and interpolation between the two global optima using a model with a single hidden layer.



(b) Solutions generatable by DO and interpolation between the two global optima using a model with two hidden layers.

FIGURE 6.21: Generatable solutions for $C = Odd3$ using a shallow (a) and deep (b) representation.

Figure 23 presents the plot of the generatable solutions for each building-block and the interpolation between two global optima after training the model with a single hidden layer. Figure 24 presents the same but for the model training with two hidden layers with the first hidden layer's weights initialized with the weights found when training the first layer alone. The plots correspond the to the models highlighted in red in Figure 22. The observations made are:

- L1 regularization alone does not perform a good compression of the search space at either level of representation – the generatable solutions plot has not improved at either representation level.
- Using no regularization, DO and DC can capture some important relationships, but as observable in the plot for 2 hidden layers, the generatable solutions have been overfit such that for many building blocks, only a single solution is generatable.
- Using DC and L2, shows that training the first hidden layer, the model does a good job of representing the solutions to the sub-functions as the as solutions to each sub-function are generatable with a good distribution. However, the interpolation

plot shows a noisy neighborhood as variables are being reassigned multiple times, and there exists a few peaks in the fitness function. The addition of the second hidden layers, causes a significant transformation to the first hidden layer, where it has lost its ability to generate solutions from this layer. The addition of the second hidden layer has not provide a significant improvement when compared to the first hidden layer's performance.

- Using DO+L1+L2 provides a very good representation. The 1st hidden layer allows for all solutions to the sub-functions to be generated and the interpolation OK. The addition of the second layer improves the performance of the representation. The generatable solutions is improved but most significantly the interpolation is improved with regards to both the generated solution and fitness, thus signifying the neighborhood space at the 2$^{\text{nd}}$ hidden layer is good for searching via local variation.

In Summary, DO is most important for inducing a good representation of the solution space. L1 and L2 help refine the representation but are not as influential for the Podd3 problem as there are more destructive of the relationships.

Interestingly, there is evidence that a single hidden layer is capable of representing the solutions to the building-blocks well. This is evident from the generatable solutions plot for tor the first layer. However, the representation does not provide a good compression of the solution space such that neighboring points in the latent space are meaningful as inferred from the interpolation plots. However, for the 2nd hidden layer case, the model now has the capacity to perform a proper compression such that solutions become neighbors in latent space. This is an interesting finding. The effect of multiple layers has generally been attributed to the complexity of the relationships. However, here it shows the complex relationships can be captured well by instead using many units. Even when doing so, this does not make it suitable for local variation at the latent space. The second hidden layer, however, performs this compression of the relationships so that this does occur. This is important for DO in answering the question of how to perform the layer wise method. Do does not have to be restricted to adding a single layer at a time. However, doing is still a robust method, while more units may be required to capture the relationships, initially, information will not be lost. At the addition of the second hidden layer, the layer will either perform a different function by combining these relationships into a more compressed representation.

All the experiments presented used a compression ratio of 0.8 (the size of the hidden layer is 0.8 the size of the layer below). A denoising autoencoder, such as using dropout on the input representation, is known to reduce the sensitivity of the compression ratio. To validate this, an experiment is performed using the challenging problems Podd3 and Podd4 with models that have a compression ratio of 2.0 – the size of the first hidden layer is twice the size of the input (an expansion of dimensionality). Figure 25 presents the weights of the first hidden layer using tuned L1 and L2 regularization with Drop-out only or Drop-connect only. The effect is clear. Drop-connect is unable to overcome the pressure of the regularization, as such this leads to a poor representation. Dropout performs very well, in fact using an expansion has helped improve the performance when compared to the previous weight's matrix presented earlier. Note for the parity3 function, the interpolation space would still be relatively poor. This experiment signifies that dropout is a very useful method for inducing a representation to captures the relationships between the variables. Drop-connect on the other provides no improvement compared to Drop-out alone. Combining drop-connect with drop-out appears to have no measurable effect on performance.

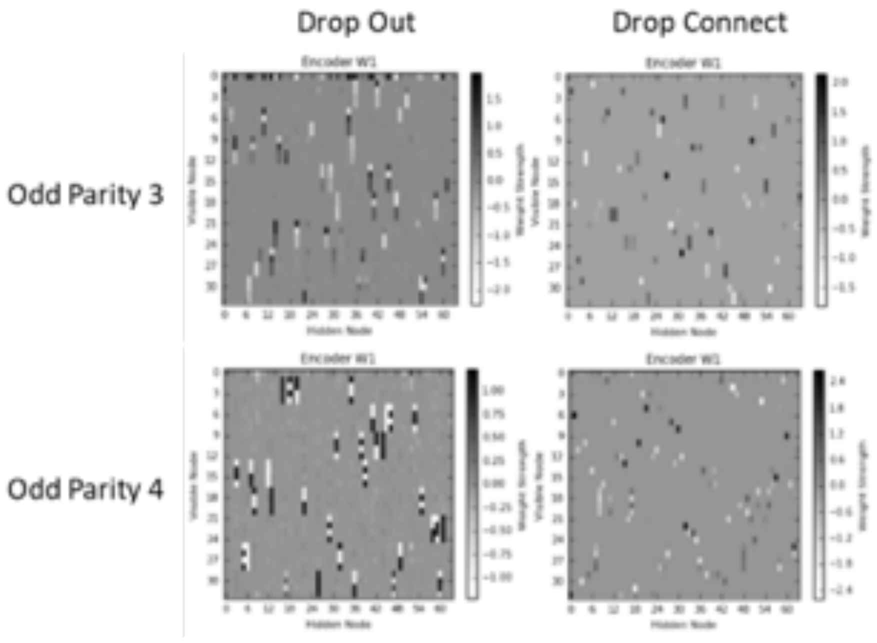


FIGURE 6.22: Argument for dropout. Encourages the representation not to learn the identity function. The identity function is still learnt when using drop connect. Parity 3 problem. First layer of the 2-layer network for an expansion of 2.

6.3.3 Summary

It is important to provide an understanding of the parameters effect on inducing representations for MIV. The technique requires a good separation of direction of variation and a good capture of the relationships that define the direction of variation. Using the theoretical optimisation problem, it is easy to verify how well the model has represented and compressed the solution space. In real-world optimisation problems, there is no access to the problem structure. Therefore, having an understanding of how regularization and denoising encourages a good representation robustly (across all problem types) provides a general rule-of-thumb for when applying to real-world optimisation problems. To conclusion taken from these experiments are:

1. The autoencoder model is very capable of representing the dependencies between variables within building blocks and separating the building blocks.

2. Regularization plays an important role, as this can reduce the number of training solutions required to induce the representation.

3. L1 is very good at separating connections and therefore encourages a function close to the identity function when many connections are required per a neuron to capture a relationship

4. DO is very good at improving the representation learnt. This is not unexpected. Dropout causes the decoder to have to decide what value to assign to a missing variable. This can only be inferred from relationships with other variables.

5. DC causes multiple connections for the representation because it is favoring a robust representation to faulty connections. If there are extra connections such that the value of the variable when generating can be inferred from multiple connections, it makes the model robust to DC. This is generally a favorable property. However, it does create a representation where many neurons have the same response (same or compliment weight vectors/filters). Thus, making it less suitable for MIV. DC is not crucial if using DO

6. The $C = odd3$ function does not necessarily require two hidden layers to represent the solutions, however, it does require to hidden layers to learn compact representation such that neighbors at the latent representation are meaningful.

7. A model that has a visually clear separation of building-blocks in the weights corresponded to a latent space that had learnt a good compression of the solution space and was able to evenly generate a distribution of solutions for each sub-function while removing incorrect solutions to the sub-functions. Thus, one can use the weights matrix (at the first layer) to guide the performance of training. This is important for real problems as the structure will be unknown.

6.4 Model Exploitation

How good the representation is dependent on how the information is exploited from the model. For MIV it is important that the representation separates the modes of variations and describes the direction of variation. Section [Representation Learning] verified that an autoencoder model could capture and separative the directions of variation. In this section experiments are performed to verify the methods for exploiting this information in order to improve a solution.

The neighborhood is defined by both the representation and the method for moving in that representation. Therefore, the performance of MIV will be dependent on the representation. For these experiments all three environments are explore for all complexity classes as each environment has different characteristics of how one can move in the space. The aim of this section is to verify both the variation made to the latent representation (dH) and the corresponding variation made to the solution representation(dX). The methods for calculating dH and dX are detailed in Chapter 4. The experiments first train a model using local optimum solutions using parameters tuned to induce a good model representation. Then for each variation method, 30 training examples are used as the solution to be improved. The MIV method is then run for each solution until no further improvements are made. The size of variations made, the fitness changes and the number of function evaluations performed are all measured for each solution. Each variation method uses the same model and the starting solutions are the same for each variation experiment. The MIV method is the changeable variable in these experiments. The aim of this section is to verify the efficiency and robustness of the latent variation methods. Efficiency is defined as the number of function evaluation performed to find a superior solution. Robustness is defined as the success of a method in different problem complexities and environments.

6.4.1 Solution Variation

The first experiment explores the methods uses to calculate the variation to make to a solution – dX. The details of the methods are described in Chapter 4. The sample methods interpret the reconstruction output from the decoded as the mean parameter for a binomial distribution. The threshold output methods interpret the reconstruction output as a binary value determined by value relative to the mid-activation value of the output unit. The partial methods calculate a change based on a comparison with the reconstructed value before a latent variation. This partial variation is then applied to the solution. The complete methods use complete reconstruction from the decoded to replace the solution. Figure Figure 6.23 presents the results for different output variation methods for all problem complexities sized 32. The figure reports the fitness change made to each solution using the method. Emodule is the environment function used for the experiments. For each problem complexity, the weights matrix of the model is presented to illustrate the clarity of the information learnt by the model. All models learnt a good representation of the problem structure. The Assign method is used to calculate dH. The results show the threshold methods significantly outperforms the sampling methods for calculating dX. There is no clear differentiation between partial method and complete method for either sampling or thresholding.

The results are further verified by performing the same experiments using the Enonlinear and Ehierarchical fitness landscapes. The results are presented in Figure Figure **??** where threshold methods for calculating dX consistently outperform sampling methods. As before, there is little evidence to suggest a differentiation between partial and complete solution updates.

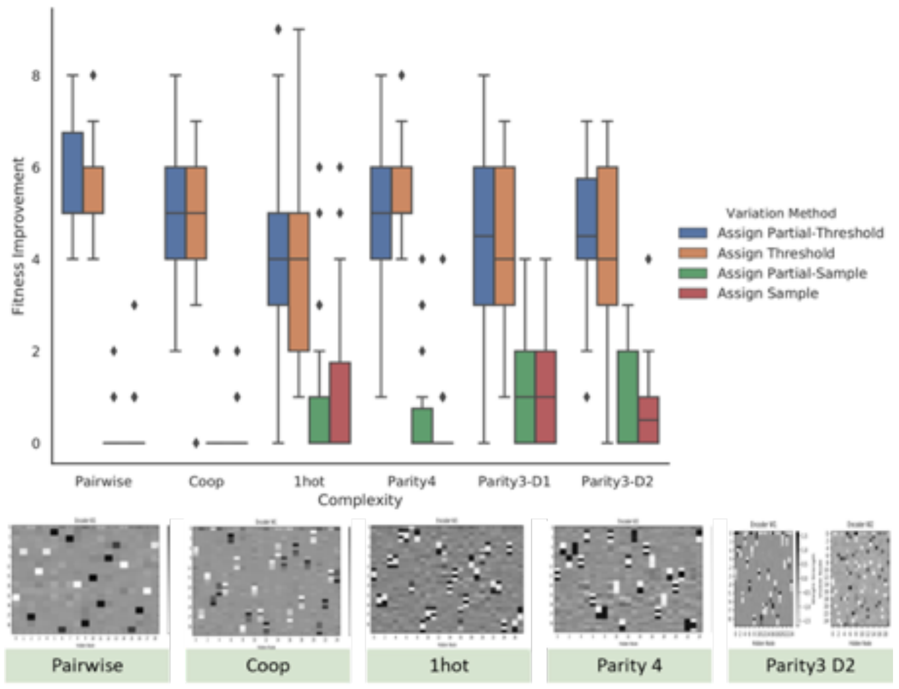


FIGURE 6.23: MIV comparison for calculating the change to a solution. Emodule problem size 32 is used. Threshold methods significantly outperform sampling methods for all problem complexities

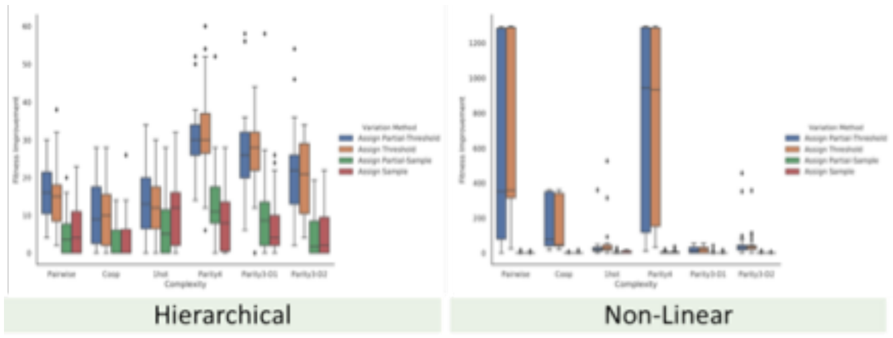


FIGURE 6.24: Model information exploitation comparison for the Encourage representation Phier and Pman problem size 32 output interpretation method

6.4.2 Latent Variation

This section verifies the method for making a change to the latent representation. As detailed in Chapter 4, the latent representation provides a new neighborhood for the solution space. This neighborhood is defined by both the representation and the variation operator that moves between different representations. All MIV methods start by encoding the current solution to provide a latent representation of the solution. The methods used to apply a variation to the encoded representation, dH, are:

1. Flip – randomly select a hidden unit and flip the activation value. For a sigmoid unit, this would reflect the activation value at the activation point 0, e.g. 0.8 is flipped to 0.2.
2. Empirical – randomly select a hidden unit and replace the activation value with an activation value of a different solution chosen at random.
3. Assign – randomly select a hidden unit and assign the activation value to an extremum of the activation range (1 or 0 if using the sigmoid function).
4. Encode-Assign – randomly select an input unit and calculate the sensitivity of a hidden unit's activation response to the change of that input unit. For the hidden units that have enough sensitivity assign the hidden activation values to an extremum of the activation range
5. Encode-Direct – randomly select an input unit and calculate the sensitivity of a hidden unit's activation response to the change of that input unit. For the hidden units that have enough sensitivity assign the hidden activation values to the direction of change calculated by encoding the change between the original input and the original input that has the variable at the opposite value.

The experimental set-up is the same as previously described in section [Solutions Variations - dX]. Sample methods are not reported here as they were previously shown to be inferior to threshold methods for calculating dX. Figure Figure 6.26 presents the fitness changes made to all complexity functions for all fitness functions. The partial-threshold method is reported as the complete threshold method has a similar performance and can therefore draw the same conclusions. The observations are:

- Empirical method has the worst performance across all complexity functions and environments.
- For the Emodule fitness environment there is insufficient evidence to differentiate between the dH methods flip, assign, encode-assign and encode-direct.
- For the Ehierarchical fitness environment calculating which hidden units to varying by encoding the response shows a significant improvement in the 1hot and coop complexities, where the direction-based method outperforms the random assignment method. On the other complexities, encode-direct has as good as performance as any other method. Assign outperforms the flip variation.
- For the Enonlinear fitness environment the differentiation between the latent variation methods is most prominent. The encode direct method, for all problem

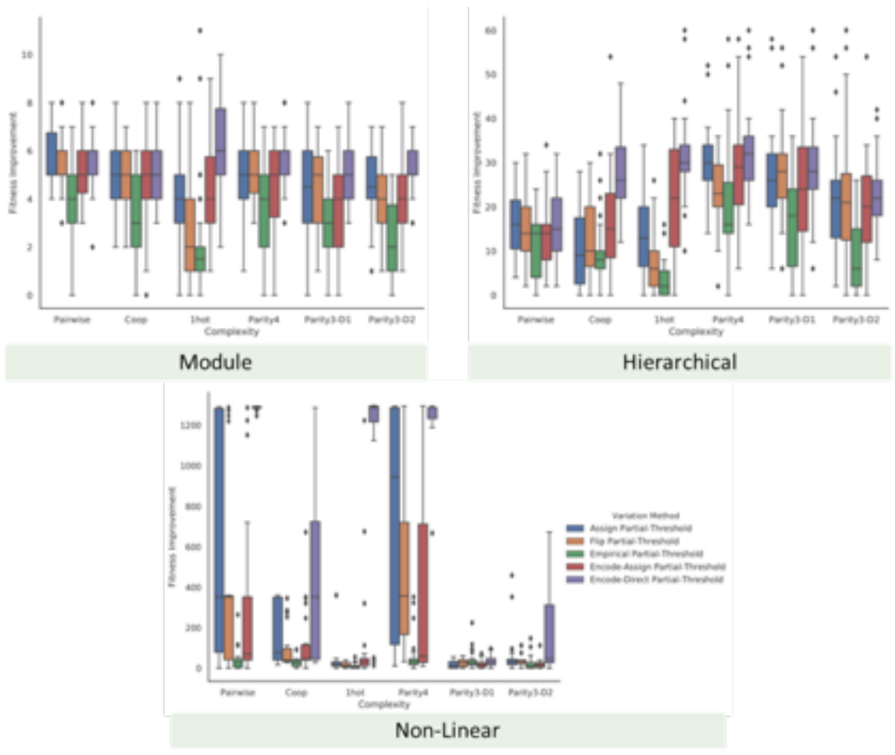


Figure 6.25: Comparison of methods for calculating the latent variation for problems Pmod Phier and Pman for problem size 32

complexities is most efficient method and is superior for the 1hot and Odd3 problem complexities.

The experiment has verified that the latent variation methods encode-assign and empirical methods are the least efficient and least robust methods for exploiting the information in the model. Figure Figure 6.26 presents the results using a problem size of 128 to verify the scalability of the latent variation method. Encode-assign and empirical latent variations are not included in the experiment as the results in Figure Figure 6.26 verify these methods as less effective. Consistently, the encode-direct method is most effective for extracting the information for MIV. Most notable is for problem complexities 1hot and parity3 complexity. Further, encode-direct is most robust over all environments, performing significantly better in the Enonlinear environment.

6.4.3 Modeling-Informed Variation

Experiments thus far have led to the conclusion that the variation method that is most robust for MIV is the latent-variation method encode-direct and the solution-variation interpretation method partial-threshold. To confirm this conclusion, Figure Figure 6.27 presents the results for a combination of the most promising method from the previous experiment on a problem size of 128 for all problem complexities and environments. The observations are:

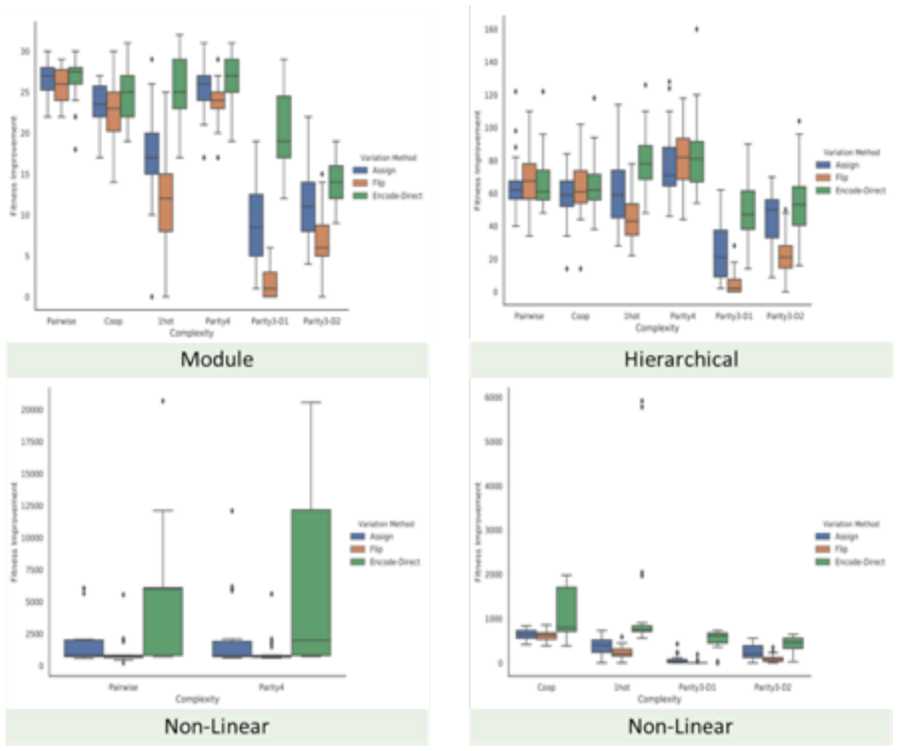


Figure 6.26: Comparison of methods for calculating the latent variation for problems Pmod Phier and Pman for problem size 128

- For the Emodule there is negligible evidence to differentiate the methods except for in P1hot case where encode-direct has greater performance.
- For the EHierarchical the encoder-direct shows a significant performance improvement compared in the Pcoop and P1hot cases.
- For Enonlinear cases encoder-direct outperforms all other methods for all complexities.
- The is insufficient evidence to differentiate the performance of threshold and partial threshold method.

6.4.4 Variation Size

In this section the changes made to a solution are explored in more detail. Measured during the previous experiment was the size of the change calculated for each MIV step. The variations are categorized according to if there provided a positive (good) fitness change, a negative (bad) fitness change or the change had a neutral effect on the fitness. Figure Fig:Eval:VarSize presents a comparison of the variations made using different output methods for calculating the change to a solution for a problem size of 128. The frequency of good, neutral and bad variations is normalized by the number of function evaluations performed. This provides a measure of efficiency for each method.

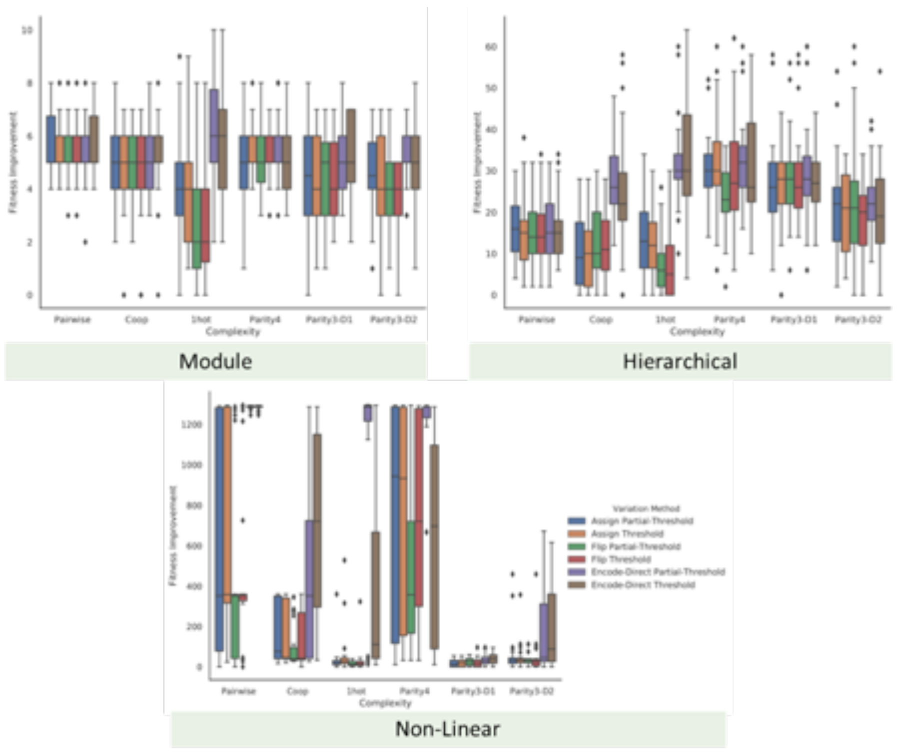


FIGURE 6.27: Comparison of the best performing dx and dh combinations on the Pmodule Phieratchical and Pnonlinear problem size 32

It is clear to observe why sampling methods perform poorly. The size of possible variation is much greater than the threshold method, however, the influence of a change at the hidden layer is nullified. For instance, a change to the hidden representation that creates a change to the decoding will provided a positive fitness improvement if using the threshold method can be destroyed if one unit is sampled incorrectly. In general partial-threshold is more efficient that the threshold method.

In the parity 3 cases, sampling methods perform as efficiently as the threshold methods. This can be explained by the activation response presented in Figure Figure 6.29. For each problem complexity a sample of 10 solutions are forward propagated through the autoencoder model and the activation values measured. Each response is plotted to present a visualisation of the average response. The model used for the parity 3 function is unique in that the output response is saturated. Therefore, sampling the output will have a similar behaviour to the threshold. This is unrealistic in expectation of the model in practice

Figure Figure **??** presents the results of variation perform for all complexities and environments using different latent variation methods on a problem size of 128. Encode-assign and empirical are not reported due to the poor performance observed in section [Latent variations]. As partial-threshold and threshold output methods have shown negligible differentiate in performance the partial-threshold method is only reported in the results. Sampling and partial sampling are removed due to their poor performance.

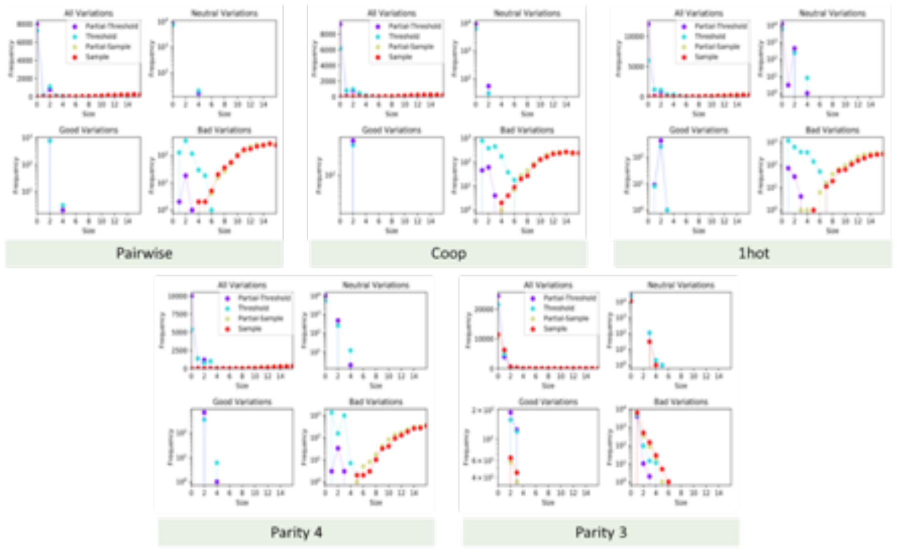


Figure 6.28: Variations performed during MIV with different output (dx) methods for all complexity types. Sampling performs poorly except on parity 3. Scaled by the total number of wasted function evaluations performed (FE).

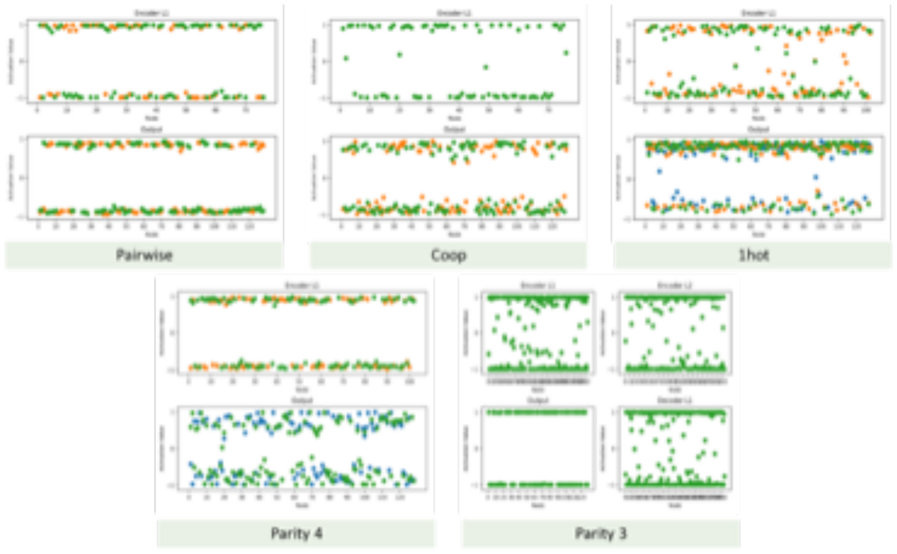


Figure 6.29: Activation response for model training of different complexities. Parity 3 model is unique with output activation's completely saturated. Therefore, calculating dx by sampling performs well.

Good, neutral and bad variations. Generally, encode-direct is less efficient due to performing multiple fitness evaluations per a single MIV step. However, when observing one-hot, the efficiency of the methods appears competitive the encode-direct method performs many more variations of size 2 compared to the other methods.

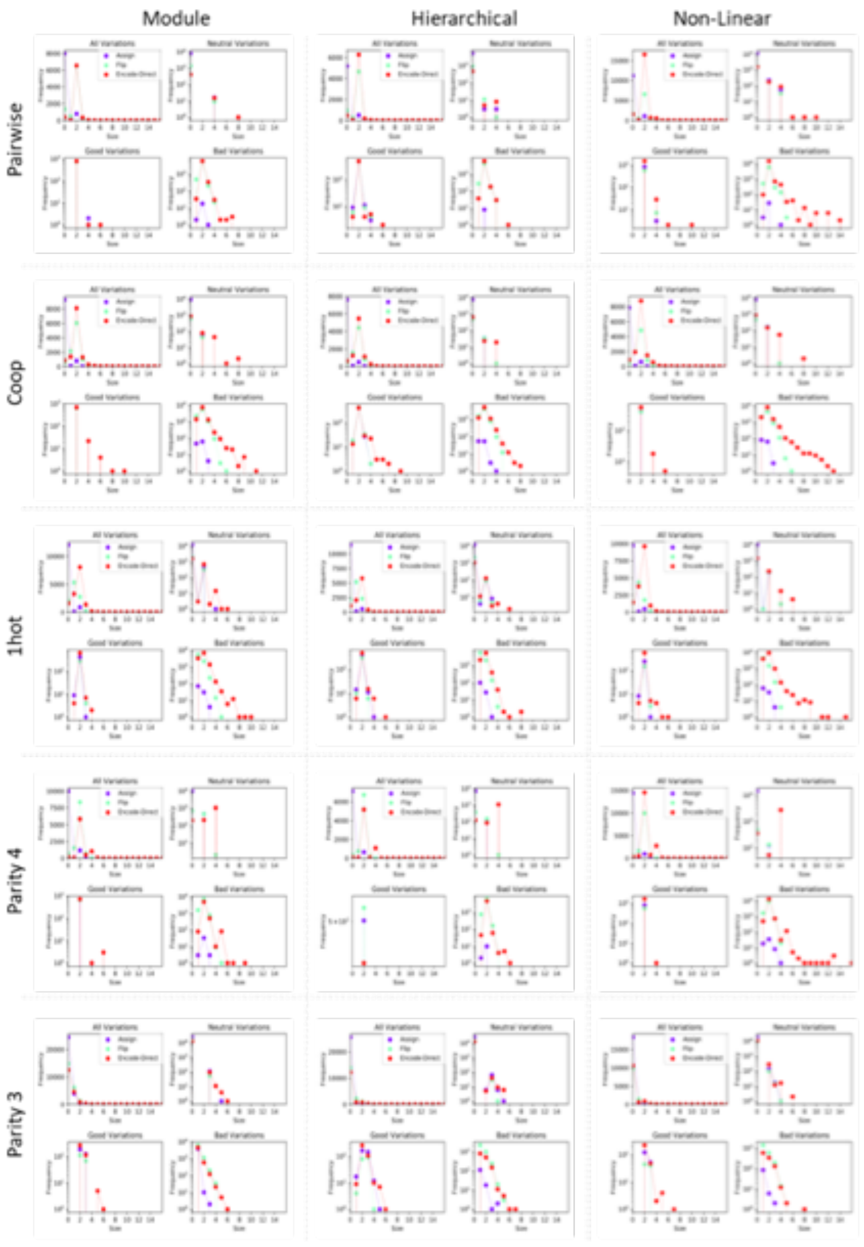


FIGURE 6.30: Performance comparison of latent variation on the variation made to the problem.

6.4.5 Summary

The Emodule fitness environment doesn't provide a particularly challenging environment to search in. The accuracy of the exploitation appears to be insensitive. This suggest that this type of landscape is easy if we have a good representation of the solution. The EHierarchial fitness environment provides a similar challenge to the Emodule environment at the first layer, the difference is that the combination of solutions relative only to one other sub-function. It is for this reason that a similar performance is observed. However, there is a clear differential in the performance for coop overlap. This is likely due to the non-orthogonal variation that is present at the first level of variation as the state of a sub-function can be dependent on states of two other sub-functions. The Enonlinear fitness environment requires an accurate representation of the sub-function such that all solutions to the sub-function can be generated. Further it requires that all solutions to the sub-function are accessible, within the neighborhood defined by the change to latent representation. This environment provides a stronger differentiation of the methods that are unable to explore the neighborhood precisely. It is for this reason that the encode-direct method performs better, as the variation at dH is calculated rather that a single unit variation. The 1hot complexity provides a challenging relationship to perform variation. There are three filters that can learnt to represent the relationship of a single sub-function. Not all three are required to accurately represent the sub-function, but different combinations of the two create different neighborhoods for the sub-function. For example: The three filters that area used to represent the relationships are ++–, +-+- and +–+. If the sub-function is represented by the first two filters that at solution 1000, solution 0100 and 0010 are local neighbors. If the sub-function is represented by the first and third filter then at solution 1000, solution 0100 and 0001 are neighbors. Therefore, making local changes to the encoded representation can exploit the representation efficiently. The encoder-direct methods, which allows for a change to multiple hidden units in a single variation can overcome this problem.

6.5 Multi-Level Performance

We have seen in the experiments for representation learning that adding an additional hidden layer improves the representation of the lower level. As such, MIV may have not performed optimally. Performing variation only ever at the bottle neck representation is likely not to utilize lower level representations. This section investigates the performance of DO in utilizing this information by performing MIV bottom-up and top-down. DO is a product of its own dynamics. The ability to exploit the problems structure efficiently has a direct impact on the representation learnt at the next layer. The addition of a hidden layer provides to the model has many effects on the performance of DO. The primary reason for adding a hidden layer to the model is to increase the models capacity. However, DO is a product of its own dynamics. As such, an addition of a hidden layers also adds a level of robustness to the algorithm. Empirically, this section shows how an addition of a hidden layer provides:

- A compression pressure which encourages lower levels to adapt their representation to a more efficient representation
- Combining neurons that have the same relationship but and are therefore not amenable to single variations at the hidden layer.

This provides a robustness to the hyperparameter tuning of the compression ratio.

Evidence: DO constructing a deep representation of the problem structure. Methods: Use the hierarchical TP, pairwise and coop. pairwise shows do not need a deep model but we can still encourage the representation. Coop shows we do no need a deep model.

Measurements: Weights, Size of the variation made, Solutions available, Algorithm Fitness over time, Convergence.

6.5.1 Model Update

When retaining the same model, after training, the model is used for generation. However, when retiaining the generation model, the generation model only uses the information in the traininied model after transition. Figure Figure 6.32

6.5.2 Multi-Level Variation

Each level of representation in the deep neural network provides a different neighbourhood for the solutions space.

Layerwise Top-Down Bottom-Up

What is the section about: - A naïve method for MIV is to use only the bottleneck layer. However, large variations can be favoured due to the size of the change, yet during this variation small combinations of variables can be destroyed. Therefore, throughout the development of DO it has been found beneficial to repair the solution. This is simply performing hill-climbing on the solution representation using the primitive variation operator. - This leads to the question, why only repair at the solution level representation. One can traverse the solution representation, defined by the model, and hill-climb at each representation level. This covers the repair procedure while adding additional potential as repair can be performed at intermediate levels of representation. - Because a solution is only replaced if it outperforms its previous fitness after the complete MIV procedure, repair can to a solution can occur to a solution at the next iteration. Take for example the scenario described previously, using a bottom up approach would not provide a repair at the lower level. Yet at the next iteration, the lower level is first search, thus providing the repair to the solution. - The two directions of traversal of the deep representation are possible. The section demonstrates the performance of the top-down and bottom-up traversal methods. To demonstrate the performance, difference in fitness evolution, solution trajectory, fitness evaluations to global optimum See that there is no need for the repair really for these problems.

Results

Conclusion The performance difference between the top-down and bottom-up traversal methods do not demonstrate a significant difference on the theoretical problem.

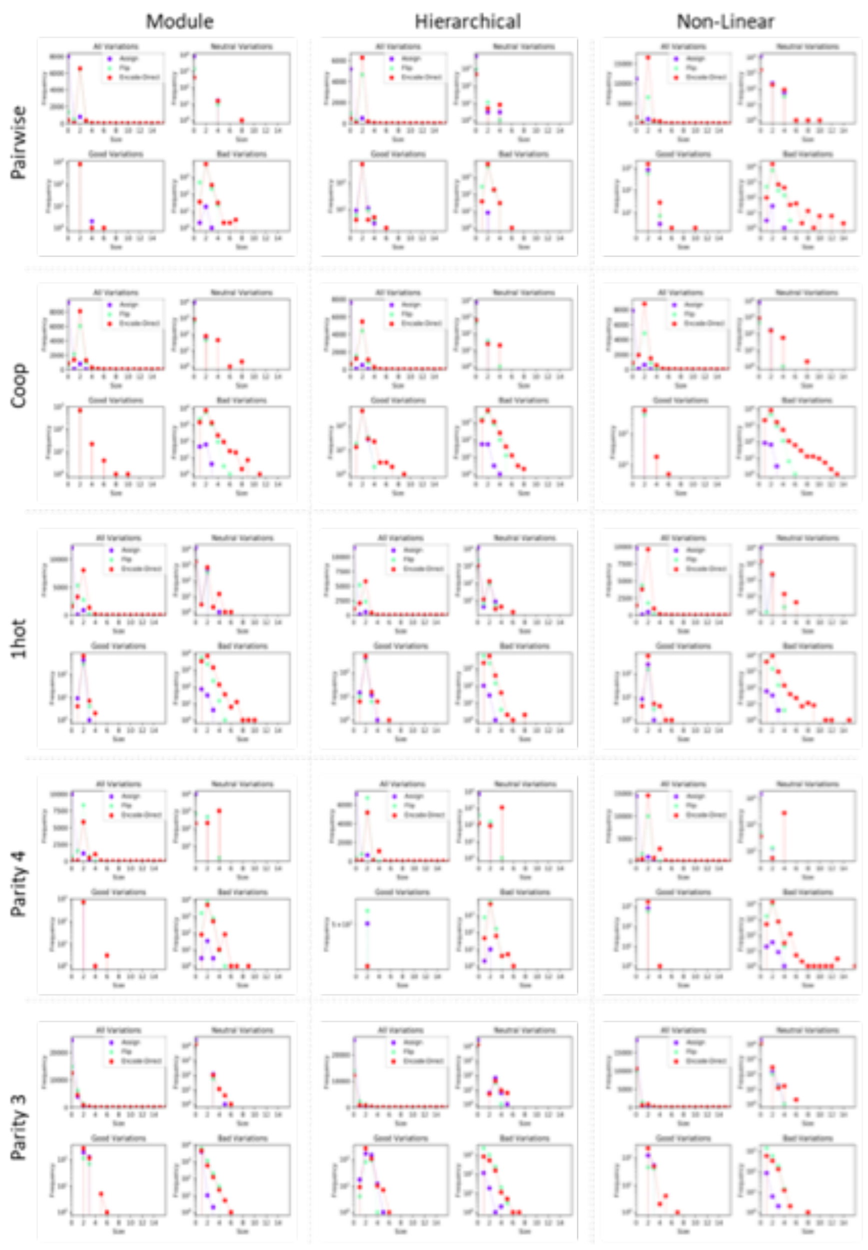


Figure 6.31: Separating the model for training and generating ensures improves the robustness of exploiting the information.

6.5.3 MIG Initialisation

Encoding and sampling - Enable a combination of MIG and MIV in a single algorithm.

- Using MIG for bottom-up produces the risk of not repairing the solution at the lower-level. - Using MIG for top-down makes more sense as the lower level will be repaired. However, if MIG is performing poorly, moving in the latent representation space may be easy and non-meaningful. Thus solutions with important information regarding variable combinations get overwritten. Helps converge better – manifold 1hot 128 Show trajectories

6.5.4 Layer-wise Learning

The hierarchical construction investigate the ability to re-scale the building-blocks.

$C = NO$ Don't need a deep representation, the model can compress into a single layer Can encourage a deep representation never-the-less. Useful for dynamic optimisation as demonstrated in Chapter 6. Results to get: - Scaling pairwise – shallow - Visualisation of the weights

Overlap Need a deep representation, a shallow representation fails to combine modules at the lower level. A deep representation represents the multi-level structure well.

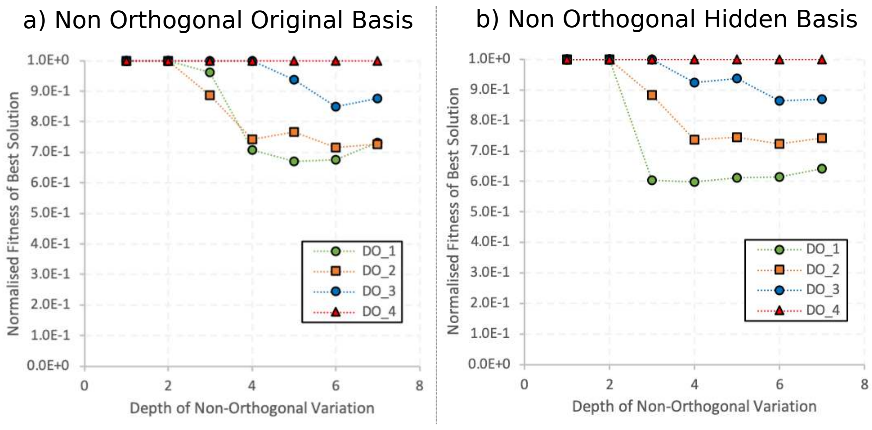


FIGURE 6.32: Limiting the depth of DO fails to solve hierarchical problems with overlapping sub-functions.

Odd 3 - Scaling behavior of parity 3 using a shallow network for the module problem – representation. - o Simply fails – fails to represent - Visualisation of the weights and fitness progress

6.5.5 Layer-wise Stacking

-Only meaningful when using bottom-up approach -Causes to learn a deep representation of the structure – HIFF becomes deep -Useful for forcing the model to maintain the variation structure. from the original solution space. -Forced to learn a deep structure

Results to get: - Visualisation of the weights and fitness progress - Redo the experiments for DO

Discussion

Method of separating variables at the natural joints is widely used for simplifying the search for global function. This makes searching the space and further dimensional reduction efficient. If the direction of variation is orthogonal, then there is no requirement to change the basis of the variation from the original representation. Therefore, no need for a deep representation. If the direction of variation is non-orthogonal in the original solution representation, then a non-linear representation is required to transform the representation such that orthogonal variation operators can succeed.

6.6 Summary

We use theoretical problems with known and controllable structure to gain a deeper understanding into how DO works. By linking optimisation with the machine learning community enables us to use advanced tools that are used for accessing the performance of neural networks on their subsequent task. We utilize these approaches and adapt them to optimisation problems to probe the performance of DO. We first demonstrate that validity of this by applying to the theoretical problems. We know the structure and therefore can confirm if the tools that attempt to understand what the model has learnt correlates with the known structure. We then apply this to common benchmark instances. We want to access DO performance in terms of quality of solution found, but also what structure the model has learnt, how useful this structure is, and can this information be used to help develop out understanding further about how to design optimisation algorithms.

Chapter 7

Application of Deep Optimisation

this aids researcher into further developing their understanding of how and why the algorithm works, its application to industry is unknown as the theoretical problems are idealised.

Therefore, DO is unlikely to be competitive on benchmark problems that are randomly generated. Unfortunate this can be a large component that is used to determine the performance of an algorithm. Generally, interested in a problem class, that contains random structure instance. However, we note that 'real world' problems are not random. Further, they generally are composed of a multitude of optimisation domains. In simple cases we can cast the 'real-world' problem into a specific domain and utilise the domain specific algorithm. However, this approximation may not be appropriate in the presence of multi-component problem etc. Further, the real world is adaptive. Problem definitions change and constraints may come and go. Thus the complexity of casting the 'real-world' optimisation time several times may be infeasible.

7.1 Knapsack

7.2 TSP

7.3 Continuous Optimisation - Griewank function

7.4 Repeated Optimisation (Dynamic Optimisation)

7.5 Multi-Objective Optimisation

7.6 Discussion

Chapter 8

Conclusions

8.1 What has been done

8.2 Main Conclusions

This thesis connects the domain of MBOAs and deep learning by demonstrating state-of-the-art results. This is achieved by emulating the theory of ETI, providing an interpretation that is aligned with recent thinking in biological evolution Watson and Szathmáry (2016); Laland et al. (2015). In doing so, we provide the founding understanding for how a deep neural network can be used within the domain of evolutionary computing. DO thus demonstrates that deep learning methods can be used for optimisation problems and provides results that are distinct from existing methods. More importantly, whilst this implementation of DO uses a specific deep learning model and particular model-learning methods, it opens up the domain of combinatorial optimisation to exploitation by the rich toolset of modern deep learning methods. This suggests numerous avenues for further investigation, transferring state-of-the-art deep learning methods into the domain of MBOAs.

Appendix A

Stuff

The following gets in the way of the text....

A.1 Parameters used for the experiments

P3, hBOA and LTGA used the defualt parameters for constructing and exploiting the model. More information can be found in cite(goldman code). DO's default parameters are listed in Table **??**.

A.2 Population Parameter Search Algorithm

Algorithm A.2 is the search method used to tune the population size of an MBOA. The criteria for the search is to find a minimum popuation size such that no failures occur in all repeated runs. Due to the stochasticity of the algorithm, and or problem, it is possible that smaller populations find a solution while in other instances it was not sufficient. Thus why a distribution of algorithm runs is used to determine the sizing of a population.

FIGURE A.1: Population Size Setting

InitialPopulationSize set to 10 P in ProblemSize AdditionalSize = PopulationSize / 10 Number of failures ¿ 0 **Run Optimizer with InitialPopulationSize + AdditionalSize** Failures ¿ 0 AdditionalSize += AdditionalSize Record algorithm results InitialPopulationSize = (InitialPopulationSize + AdditionalSize) / 2

Table A.1: Parameters used for the experiments

Order	Size	LTGA Population	HBOA Population	DO L1
6*0	16			
	32			
	64			
	128			
	256			
	512			
6*1	16			
	32			
	64			
	128			
	256			
	512			
6*2	16			
	32			
	64			
	128			
	256			
	512			

Bibliography

Uwe Aickelin, Edmund K Burke, and Jingpeng Li. An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. *Journal of the Operational Research Society*, 58(12):1574–1585, 2007.

Lee Altenberg et al. The evolution of evolvability in genetic programming. *Advances in genetic programming*, 3:47–74, 1994.

David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study.* Princeton university press, 2006.

Shumeet Baluja. Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science, 1994.

Shumeet Baluja. Deep learning for explicitly modeling optimization landscapes. *arXiv preprint arXiv:1703.07394*, 2017.

Shumeet Baluja and Scott Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1997.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Hans-Georg Beyer. An alternative explanation for the manner in which genetic algorithms operate. *BioSystems*, 41(1):1–15, 1997.

Peter AN Bosman and Dirk Thierens. *Linkage information processing in distribution estimation algorithms*, volume 1999. Utrecht University: Information and Computing Sciences, 1999.

Peter AN Bosman and Dirk Thierens. The roles of local search, model building and optimal mixing in evolutionary algorithms from a bbo perspective. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 663–670, 2011.

Justin Boyan and Andrew W Moore. Learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research*, 1(Nov):77–112, 2000.

Pratik Prabhanjan Brahma, Dapeng Wu, and Yiyuan She. Why deep learning works: A manifold disentanglement perspective. *IEEE transactions on neural networks and learning systems*, 27(10):1997–2008, 2015.

Rainer E Burkard, Stefan E Karisch, and Franz Rendl. Qaplib–a quadratic assignment problem library. *Journal of Global optimization*, 10(4):391–403, 1997.

Lawrence Cayton. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 12(1-17):1, 2005.

Josu Ceberio, Ekhine Irurozki, Alexander Mendiburu, and Jose A Lozano. A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. *IEEE Transactions on Evolutionary Computation*, 18(2):286–300, 2013.

Jen-Hao Chang, Chung-Hsiang Hsueh, Hsuan Lee, Tian-Li Yu, and Tsung-Yu Ho. A test function with full controllability over overlapping and confliction between sub-problems. 2011.

Deborah Charlesworth, Nicholas H Barton, and Brian Charlesworth. The sources of adaptive variation. *Proceedings of the Royal Society B: Biological Sciences*, 284(1855): 20162864, 2017.

Ping-Lin Chen, Chun-Jen Peng, Chang-Yi Lu, and Tian-Li Yu. Two-edge graphical linkage model for dsmga-ii. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 745–752, 2017.

Si-Cheng Chen and Tian-Li Yu. Difficulty of linkage learning in estimation of distribution algorithms. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 397–404, 2009.

Wei-Ming Chen, Chung-Yu Shao, Po-Chun Hsu, and Tian-Li Yu. A test problem with adjustable degrees of overlap and conflict among subproblems. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, page 257–264, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311779.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Alexander W Churchill, Siddharth Sigtia, and Chrisantha Fernando. A denoising autoencoder that guides stochastic search. *arXiv preprint arXiv:1404.1614*, 2014a.

Alexander W Churchill, Siddharth Sigtia, and Chrisantha Fernando. A denoising autoencoder that guides stochastic search. *arXiv preprint arXiv:1404.1614*, 2014b.

David Jonathan Coffin and Robert Elliott Smith. The limitations of distribution sampling for linkage learning. In *2007 IEEE Congress on Evolutionary Computation*, pages 364–369. IEEE, 2007.

Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

Gregory F Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

Chris Cox. *Inferring and exploiting compact models of evolutionary problem structure.* PhD thesis, University of Southampton, 2015.

Chris R Cox and Richard A Watson. Inferring and exploiting problem structure with schema grammar. In *International Conference on Parallel Problem Solving from Nature*, pages 404–413. Springer, 2014a.

Chris R Cox and Richard A Watson. Solving building block problems using generative grammar. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 341–348, 2014b.

Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)*, 45(3):1–33, 2013.

Jeremy S De Bonet, Charles Lee Isbell Jr, and Paul A Viola. Mimic: Finding optima by estimating probability densities. In *Advances in neural information processing systems*, pages 424–430, 1997.

Kenneth A De Jong. *Evolutionary Computation: A Unified Approach.* MIT Press, 2006.

Kenneth Alan De Jong. Analysis of the behavior of a class of genetic adaptive systems. Technical report, 1975.

Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2-4):311–338, 2000.

Kalyanmoy Deb and David E. Goldberg. Analyzing deception in trap functions. In L. DARRELL WHITLEY, editor, *Foundations of Genetic Algorithms*, volume 2 of *Foundations of Genetic Algorithms*, pages 93 – 108. Elsevier, 1993.

Kalyanmoy Deb and David E Goldberg. Sufficient conditions for deceptive and easy binary functions. *Annals of mathematics and Artificial Intelligence*, 10(4):385–408, 1994.

Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

Gal Elidan, Noam Lotner, Nir Friedman, and Daphne Koller. Discovering hidden variables: A structure-based approach. In *Advances in Neural Information Processing Systems*, pages 479–485, 2001.

Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

D. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. 1987.

David E Goldberg, Kalyanmoy Deb, James H Clark, et al. Genetic algorithms, noise, and the sizing of populations. *COMPLEX SYSTEMS-CHAMPAIGN-*, 6:333–333, 1992.

David E Goldberg, Kalyanmoy Deb, and Dirk Thierens. Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, 32(1):10–16, 1993.

David E Goldberg, Bradley Korb, and Kalyanmoy Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, 3(5):493–530, 1989.

David E Goldberg, Jon Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

DE Goldberg. Genetic algorithms in search, optimization, and machine learning, addison-wesley, reading, ma, 1989. *NN Schraudolph and J*, 3(1), 1989.

Brian W. Goldman and William F. Punch. Parameter-less population pyramid. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, page 785–792, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326629.

Brian W Goldman and William F Punch. Fast and efficient black box optimization using the parameter-less population pyramid. *Evolutionary computation*, 23(3):451–479, 2015.

Pierre Hansen, Nenad Mladenović, and José A Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

Georges Harik et al. Linkage learning via probabilistic modeling in the ecga. 1999a.

Georges R Harik. Finding multimodal solutions using restricted tournament selection.

Georges R Harik, Fernando G Lobo, and David E Goldberg. The compact genetic algorithm. *IEEE transactions on evolutionary computation*, 3(4):287–297, 1999b.

Georges Raif Harik. Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. 1997.

Georges Raif Harik. Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. 1998.

Mark Hauschild and Martin Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and evolutionary computation*, 1(3):111–128, 2011.

D O HEBB. *THE ORGANIZATION OF BEHAVIOUR.* 1961.

Robert B Heckendorn, Soraya Rana, and Darrell Whitley. Test function generators as embedded landscapes. *Foundations of Genetic Algorithms*, 5:183–198, 1999.

Francisco Herrera, Manuel Lozano, and Ana M Sánchez. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems*, 18(3):309–338, 2003.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012a.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012b.

Christian Höhn and C. Reeves. Are long path problems hard for genetic algorithms? In *PPSN*, 1996.

John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

John J Hopfield and David W Tank. "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

Jeffrey Horn, David E Goldberg, and Kalyanmoy Deb. Long path problems. In *International Conference on Parallel Problem Solving from Nature*, pages 149–158. Springer, 1994.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Shih-Huan Hsu and Tian-Li Yu. 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*, pages 519–526, 2015.

David Iclanzan. Hierarchical allelic pairwise independent functions. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 633–640, 2011.

David Iclanzan and Dan Dumitrescu. Overcoming hierarchical difficulty by hill-climbing the building block structure. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1256–1263, 2007.

Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. *Advances in neural information processing systems*, 21:769–776, 2008.

Hillol Kargupta. The gene expression messy genetic algorithm. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 814–819. IEEE, 1996.

Stuart A Kauffman et al. *The origins of order: Self-organization and selection in evolution.* Oxford University Press, USA, 1993.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

Philip Kilby, John Slaney, Sylvie Thiébaux, Toby Walsh, et al. Backbones and backdoors in satisfiability. In *AAAI*, volume 5, pages 1368–1373, 2005.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Joshua D Knowles, Richard A Watson, and David W Corne. Reducing local optima in single-objective problems by multi-objectivization. In *International conference on evolutionary multi-criterion optimization*, pages 269–283. Springer, 2001.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017.

Kevin Laland, John Odling-Smee, and Scott Turner. The role of internal and external constructive processes in evolution. *The Journal of physiology*, 592(11):2413–2422, 2014.

Kevin N Laland, Tobias Uller, Marcus W Feldman, Kim Sterelny, Gerd B Müller, Armin Moczek, Eva Jablonka, and John Odling-Smee. The extended evolutionary synthesis: its structure, assumptions and predictions. *Proceedings of the Royal Society B: Biological Sciences*, 282(1813):20151019, 2015.

Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. Sparse deep belief net model for visual area v2. *Advances in neural information processing systems*, 20:873–880, 2007.

Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

Claudio F Lima, Martin Pelikan, Kumara Sastry, Martin Butz, David E Goldberg, and Fernando G Lobo. Substructural neighborhoods for local search in the bayesian optimization algorithm. In *Parallel Problem Solving from Nature-PPSN IX*, pages 232–241. Springer, 2006.

Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

Michele Lombardi, Michela Milano, and Andrea Bartolini. Empirical decision model learning. *Artificial Intelligence*, 244:343–367, 2017.

Samir W Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, Citeseer, 1995.

Jean P Martins and Alexandre CB Delbem. Pairwise independence and its impact on estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 27: 80–96, 2016.

Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *arXiv preprint arXiv:2003.03600*, 2020.

Efrén Mezura-Montes and Carlos A Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

Rob Mills. *How micro-evolution can guide macro-evolution: Multi-scale search via evolved modular variation.* PhD thesis, University of Southampton, 2010.

Rob Mills, Thomas Jansen, and Richard A Watson. Transforming evolutionary search into higher-level evolutionary search by capturing problem structure. *IEEE Transactions on Evolutionary Computation*, 18(5):628–642, 2014.

Rob Mills and Richard A Watson. Multi-scale search, modular variation, and adaptive neighbourhoods. *Author's Original*, 2011.

Marvin Minsky and Seymour Papert. Perceptrons. 1969.

Melanie Mitchell, Stephanie Forrest, and John H Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. *Ann Arbor*, 1001:48109.

Heinz Mühlenbein and Thilo Mahnig. Fda-a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary computation*, 7(4): 353–376, 1999.

Heinz Mühlenbein and Gerhard Paass. From recombination of genes to the estimation of distributions i. binary parameters. In *International conference on parallel problem solving from nature*, pages 178–187. Springer, 1996.

Una-May O'Reilly and Franz Oppacher. The troubling aspects of a building block hypothesis for genetic programming. In *Foundations of genetic algorithms*, volume 3, pages 73–88. Elsevier, 1995.

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity.* Courier Corporation, 1998.

G Pavai and TV Geetha. A survey on crossover operators. *ACM Computing Surveys (CSUR)*, 49(4):1–43, 2016.

Martin Pelikan. Nk landscapes, problem difficulty, and hybrid evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 665–672, 2010.

Martin Pelikan and David E Goldberg. Escaping hierarchical traps with competent genetic algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 511–518. Morgan Kaufmann Publishers Inc., 2001.

Martin Pelikan and David E. Goldberg. Hierarchical boa solves ising spin glasses and maxsat. In *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation: PartII*, GECCO'03, page 1271–1282, Berlin, Heidelberg, 2003a. Springer-Verlag. ISBN 3540406034.

Martin Pelikan and David E Goldberg. A hierarchy machine: Learning to optimize from nature and humans. *Complexity*, 8(5):36–45, 2003b.

Martin Pelikan and David E Goldberg. Hierarchical bayesian optimization algorithm. In *Scalable optimization via probabilistic modeling*, pages 63–90. Springer, 2006.

Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. Boa: The bayesian optimization algorithm. In *Proceedings of the genetic and evolutionary computation conference GECCO-99*, volume 1, pages 525–532, 1999.

Martin Pelikan, David E Goldberg, and Shigeyoshi Tsutsui. Hierarchical bayesian optimization algorithm: toward a new generation of evolutionary algorithms. In *SICE 2003 Annual Conference (IEEE Cat. No. 03TH8734)*, volume 3, pages 2738–2743. IEEE, 2003.

Martin Pelikan, Mark W Hauschild, and Dirk Thierens. Pairwise and problem-specific distance metrics in the linkage tree genetic algorithm. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1005–1012, 2011.

Martin Pelikan and Heinz Mühlenbein. The bivariate marginal distribution algorithm. In *Advances in Soft Computing*, pages 521–535. Springer, 1999.

Massimo Pigliucci and Gerd B Müller. *Evolution-the extended synthesis.* The MIT Press, 2010.

Jean-Yves Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63(3):337–370, 1996.

Malte Probst. Denoising autoencoders for fast combinatorial black box optimization, 2015.

Malte Probst and Franz Rothlauf. Deep boltzmann machines in estimation of distribution algorithms for combinatorial optimization. *arXiv preprint arXiv:1509.06535*, 2015.

Adam Prugel-Bennett. Finding critical backbone structures with genetic algorithms. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1343–1348, 2007.

Nestor V Queipo, Raphael T Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P Kevin Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1):1–28, 2005.

Elizabeth Radetic and Martin Pelikan. Spurious dependencies and eda scalability. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 303–310, 2010.

Salah Rifai, Yoshua Bengio, Yann Dauphin, and Pascal Vincent. A generative process for sampling contractive auto-encoders. *arXiv preprint arXiv:1206.6434*, 2012.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Icml*, 2011.

Franz Rothlauf. Representations for genetic and evolutionary algorithms. In *Representations for Genetic and Evolutionary Algorithms*, pages 9–32. Springer, 2006.

Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

Sancho Salcedo-Sanz. A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer science review*, 3(3):175–192, 2009.

Ralf Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278, 1996.

Roberto Santana. Gray-box optimization and factorized distribution algorithms: where two worlds collide, 2017.

Roberto Santana, Pedro Larrañaga, and Jose A Lozano. Protein folding in simplified models with estimation of distribution algorithms. *IEEE transactions on Evolutionary Computation*, 12(4):418–438, 2008.

Kumara Sastry and David E Goldberg. Designing competent mutation operators via probabilistic model building of neighborhoods. In *Genetic and Evolutionary Computation Conference*, pages 114–125. Springer, 2004.

David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26):1792, 1975.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Herbert A Simon. The sciences of the artificial. 1969.

John Maynard Smith and Eörs Szathmáry. *The major transitions in evolution.* Oxford University Press, 1997.

Emilie C Snell-Rood. Selective processes in development: implications for the costs and benefits of phenotypic plasticity. *Integrative and comparative biology*, page ics067, 2012.

Dirk Thierens. Analysis and design of genetic algorithms. 1995.

Dirk Thierens. Scalability problems of simple genetic algorithms. *Evolutionary computation*, 7(4):331–352, 1999.

Dirk Thierens. The linkage tree genetic algorithm. In *International Conference on Parallel Problem Solving from Nature*, pages 264–273. Springer, 2010.

Dirk Thierens and Peter A.N. Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, page 617–624, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305570.

Dirk Thierens and Peter AN Bosman. Hierarchical problem solving with the linkage tree genetic algorithm. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 877–884, 2013.

Dirk Thierens and David E Goldberg. Mixing in genetic algorithms.

Miwako Tsuji, Masaharu Munetomo, and Kiyoshi Akama. A crossover for complex building blocks overlapping. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, page 1337–1344, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595931864.

Tobias Uller, Armin P Moczek, Richard A Watson, Paul M Brakefield, and Kevin N Laland. Developmental bias and evolution: A regulatory network perspective. *Genetics*, 209(4):949–966, 2018.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

Riccardo Volpato and Guangyan Song. Active learning to optimise time-expensive algorithm selection, 2019.

Ky Khac Vu, Claudia D'Ambrosio, Youssef Hamadi, and Leo Liberti. Surrogate-based methods for black-box optimization. *International Transactions in Operational Research*, 24(3):393–424, 2017.

Günter P Wagner and Lee Altenberg. Perspective: complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, 1996.

Shih-Ming Wang, Jie-Wei Wu, Wei-Ming Chen, and Tian-Li Yu. Design of test problems for discrete estimation of distribution algorithms. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 407–414, 2013.

Richard A Watson. *Compositional evolution: interdisciplinary investigations in evolvability, modularity, and symbiosis*. PhD thesis, Brandeis University, 2002.

Richard A Watson. *Compositional evolution: the impact of sex, symbiosis and modularity on the gradualist framework of evolution*. Mit Press, 2006.

Richard A Watson, Gregory S Hornby, and Jordan B Pollack. Modeling building-block interdependency. In *International Conference on Parallel Problem Solving from Nature*, pages 97–106. Springer, 1998.

Richard A Watson, Rob Mills, and Christopher L Buckley. Transformations in the scale of behavior and the global optimization of constraints in adaptive networks. *Adaptive Behavior*, 19(4):227–249, 2011.

Richard A Watson, Rob Mills, CL Buckley, Kostas Kouvaris, Adam Jackson, Simon T Powers, Chris Cox, Simon Tudge, Adam Davies, Loizos Kounios, et al. Evolutionary connectionism: algorithmic principles underlying the evolution of biological organisation in evo-devo, evo-eco and evolutionary transitions. *Evolutionary biology*, 43(4): 553–581, 2016.

Richard A Watson and Eörs Szathmáry. How can evolution learn? *Trends in ecology & evolution*, 31(2):147–157, 2016.

Richard A Watson, Günter P Wagner, Mihaela Pavlicev, Daniel M Weinreich, and Rob Mills. The evolution of phenotypic correlations and "developmental memory". *Evolution*, 68(4):1124–1138, 2014.

Thomas Weise, Raymond Chiong, and Ke Tang. Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology*, 27(5):907–936, 2012.

Thomas Weise, Markus Wagner, Bin Li, Xingyi Zhang, and Jörg Lässig. Special issue on benchmarking of computational intelligence algorithms in the applied soft computing journal. *Applied Soft Computing*, 93:106502, 2020. ISSN 1568-4946.

Stuart A West, Roberta M Fisher, Andy Gardner, and E Toby Kiers. Major evolutionary transitions in individuality. *Proceedings of the National Academy of Sciences*, 112(33): 10112–10119, 2015.

L Darrell Whitley. Fundamental principles of deception in genetic search. In *Foundations of genetic algorithms*, volume 1, pages 221–241. Elsevier, 1991.

David H. Wolpert. Ubiquity symposium: Evolutionary computation and the processes of life: What the no free lunch theorems really mean: How to improve search algorithms. *Ubiquity*, 2013(December), December 2013.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Tian-Li Yu, David E Goldberg, Kumara Sastry, Claudio F Lima, and Martin Pelikan. Dependency structure matrix, genetic algorithms, and effective recombination. *Evolutionary computation*, 17(4):595–626, 2009.

Tian-Li Yu, Kumara Sastry, and David E. Goldberg. Linkage learning, overlapping building blocks, and systematic strategy for scalable recombination. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, page 1217–1224, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930108.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Wei Zhang and Thomas G Dietterich. Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. *Journal of Artificial Intelligence Reseach*, 1:1–38, 2000.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from generative models. *arXiv preprint arXiv:1702.08396*, 2017.