

# Evolving Heterotic Gauge Backgrounds: Genetic Algorithms versus Reinforcement Learning

Steven Abel, Andrei Constantin,\* Thomas R. Harvey, and Andre Lukas

The immensity of the string landscape and the difficulty of identifying solutions that match the observed features of particle physics have raised serious questions about the predictive power of string theory. Modern methods of optimisation and search can, however, significantly improve the prospects of constructing the standard model in string theory. In this paper we scrutinise a corner of the heterotic string landscape consisting of compactifications on Calabi-Yau three-folds with monad bundles and show that genetic algorithms can be successfully used to generate anomaly-free supersymmetric  $SO(10)$  GUTs with three families of fermions that have the right ingredients to accommodate the standard model. We compare this method with reinforcement learning and find that the two methods have similar efficacy but somewhat complementary characteristics.

seems rather conservative in comparison with the latest estimates. For instance, in Ref. [3] it was shown that a single elliptically fibered four-fold gives rise to  $O(10^{272,000})$  F-theory flux compactifications. The second lesson is that the number of compactifications that match the symmetry group and the particle spectrum of the Standard Model is very large, despite representing only a tiny fraction of all consistent compactifications to four dimensions. In Ref. [4] it was argued that there are at least  $10^{23}$  and very likely up to  $10^{723}$  heterotic MSSMs, while the authors of Ref. [5] argued for the existence of a quadrillion standard models from F-theory.

## 1. Introduction

Obtaining string theory solutions that match the properties of particle physics in our world has been one of the main driving forces in string phenomenology, fuelling a significant effort in developing geometric engineering tools for constructing physically viable string models. This effort has met with only partial success due to the sheer mathematical difficulty of the problem and, to date, none of the existing string theory models allows for a proper description of particle physics. However, the model building experience of the past few decades has taught us much about the magnitude of the problem and about how (and also about how not) to approach it.

The first lesson we have learnt is that the size of the string landscape is much larger than previously thought. The famous first estimate of  $O(10^{500})$  consistent type IIB flux compactifications<sup>[1,2]</sup>

The third lesson is that these numbers are so large that traditional scanning methods cannot be used for systematic exploration. One can, of course, focus on small, accessible corners of the string landscape and this approach has been successful to some extent. For instance, in Refs. [4, 6],  $10^7$  pairs of Calabi-Yau three-folds and holomorphic bundles leading to  $SU(5)$  heterotic string models that can accommodate the correct MSSM spectrum were explicitly constructed. However, going beyond just the spectrum is both computationally challenging and may well lead to an empty set of models, depending on the required degree of similarity with the standard model. What is needed instead is a tool-set of non-systematic search methods, that can quickly detect phenomenologically rich patches of the string landscape and which can implement ‘on the fly’ a number of checks that go beyond the usual spectrum considerations.

Such methods can be developed using modern techniques of optimisation and search, and have already been implemented in a number of string contexts. Concretely, genetic algorithms (GAs) were introduced as a search method for finding string vacua with viable phenomenological properties in Ref. [7], with initial applications in the free fermionic formulation of heterotic string theory. More recently, reinforcement learning (RL) has been used in Ref. [8] to generate type IIA intersecting brane configurations that lead to standard-like models and in Refs. [9, 10] to identify  $SU(5)$  and  $SO(10)$  string GUT models, respectively, that can lead to three generation MSSMs. The landscape of type IIB flux vacua was explored in Ref. [11] using GAs and Markov chain Monte Carlo methods, while in Ref. [12] the same methods were used, as well as RL.

In the present paper we use a GA to obtain phenomenologically viable heterotic string compactifications on two Calabi-Yau three-folds with monad bundles. The same class of models has

S. Abel  
IPPP  
Durham University  
Durham DH1 3LE, UK  
E-mail: s.a.abel@durham.ac.uk

A. Constantin, T. R. Harvey, A. Lukas  
Rudolf Peierls Centre for Theoretical Physics  
University of Oxford  
Parks Road, Oxford OX1 3PU, UK  
E-mail: andrei.constantin@physics.ox.ac.uk;  
thomas.harvey@physics.ox.ac.uk; andre.lukas@physics.ox.ac.uk

© 2022 The Authors. *Fortschritte der Physik* published by Wiley-VCH GmbH. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.1002/prop.202200034

recently been explored in Ref. [10] using RL, and indeed one of the purposes of the present work is to compare the two methods. We search and obtain several hundred  $SO(10)$  models that can accommodate three standard model families of quarks and leptons. We also check during the course of the genetic evolution that the models are anomaly free and that they pass a necessary condition for bundle stability. The latter check would normally be performed at the end of the search, but recent advances in establishing simple, analytic formulae for line bundle cohomology<sup>[13–20]</sup> have made possible the inclusion of additional checks during the search.

The rest of the paper is organised as follows. In Section 2 we give a brief introduction to GAs while in Section 3 we review the construction of heterotic models on Calabi-Yau three-folds with monad bundles. In Section 4 we discuss the details of our GA implementation and in Section 5 we present the results.

## 2. The GA Technique for Seeking Perfect Models

Let us begin by giving some background and motivation for the GA. The approach of GAs (and evolutionary computation more generally) is to use concepts that are generally understood to underly evolution in order to find an optimum solution to a specified problem. Thus all GAs contain an evolving population of individual solution attempts, a “fitness” function for each individual, and a “selection and breeding process” based on their fitnesses, which applies evolutionary pressure with the aim of driving the population towards a solution. Depending on the problem at hand, success may be declared when the population contains sufficiently many “good” examples, or when one or several individuals in the population satisfies every one of a discrete set of desiderata. The present study will be of the latter kind. More specifically, we will be searching for models with properties that make them candidates for a standard model from string theory. For easy of terminology we will refer to such models as “perfect models” in the following.

The idea of evolutionary computing was developed in the 1950s, but it was formalised by Holland<sup>[21]</sup> who moulded the idea of “building blocks” into a theory of “schema”. Later reviews can be found in Refs. [22–30]. There has been growing interest in the technique within the astrophysics and particle theory community, for example in Refs. [7, 11, 31–51], partly due to its often surprising efficacy.

We will return to discuss some of the ideas that have been proposed to explain the efficacy of GAs in a moment, but let us first describe the technique itself in more detail as it pertains to the problem under discussion. The fundamental starting point of any GA is to encode the problem into a string of data, referred to as the *genotype* (each entry of which is referred to as an *allele*).

To be specific, in the present case the genotype is constructed as follows. Fixing a Calabi-Yau three-fold  $X$ , we seek monad bundles  $V$  constructed from two sums of holomorphic line bundles  $B$  and  $C$ , with ranks  $r_B$  and  $r_C$ , respectively, via the short exact sequence  $0 \rightarrow V \rightarrow B \rightarrow C \rightarrow 0$ . Each line bundle in  $B$  and  $C$  is specified by its first Chern class, which, relative to a basis of  $H^2(X)$ , is described by an integer list with length equal to the Picard number  $h = h^{1,1}(X)$  of  $X$ . This means that  $V$  is specified by  $h(r_B + r_C)$  integers. In fact, as explained below, we will be looking for bundles satisfying  $c_1(V) = 0$ , a constraint that cuts down the

number of integer parameters to  $h(r_B + r_C - 1)$ . In order to translate this into a genotype we will allow the line bundle integers to assume values only within a finite range, which can then be implemented as a binary encoding, with each integer being mapped to a binary string of fixed length  $n_B$  for  $B$  and  $n_C$  for  $C$ .

Thus the collection of all line bundle integers in  $B$  and  $C$  or, rather, the corresponding bit-string obtained by joining their binary encodings defines a particular model. The starting point of the GA is then a population of  $N_{\text{pop}}$  randomised individuals, each being represented as a string of digit alleles of length

$$\ell = h(r_B n_B + (r_C - 1)n_C). \quad (2.1)$$

For instance, for the first of our working examples we will have  $h = 2$ ,  $n_B = n_C = 3$ ,  $r_B = 6$  and  $r_C = 2$ , giving  $\ell = 42$ .

In the case of such a binary encoding there are general arguments<sup>[22,25,28]</sup> that suggest that the optimum size of the population  $N_{\text{pop}}$  depends only logarithmically on the genotype length  $\ell$ , with a formal lower bound

$$N_{\text{pop}} \gtrsim 1 + \frac{\log(-\ell / \log P_*)}{\log 2}, \quad (2.2)$$

determined by requiring that both possible values for each allele should appear in the initially randomised population with some representative probability  $P_*$ . (For example  $P_* = 0.999$ , would imply that there is a 1 in  $10^3$  probability that any individual allele contains only a 0 or a 1 in the entire population.) Typically this gives an estimate of  $N_{\text{pop}} \gtrsim 20$  that is a significant underestimate of the optimal size of the population, but it nevertheless gives an indication of the required population size. For the current study, the GA was found to perform well with  $N_{\text{pop}} \gtrsim 100$ . The GA then consists of repeated application of the following basic elements on the population:

**Selection:** Individuals are first selected from the population to make  $N_{\text{pop}}$  “breeding pairs” which will create a new generation to replace the original one. In order to do this the first step is to calculate the properties of each individual (its so-called *phenotype*). This particular step of extracting the phenotypes from all the genotypes is the true hallmark of a GA, as the relation between the genotype and the phenotype it generates, and also the correlations between different properties within the phenotype, may be very deeply hidden. This step, which is where the physics comes into play, is also often the most case-dependent and computationally-intensive part of the whole procedure. Encouragingly, however, it is the aspect that is most amenable to parallelisation.

Based on the phenotype, each individual is then assigned a fitness value, which is a function of the individual’s phenotype designed such that it increases as the model approaches a desired optimal solution. There is a wide variety of methods for assigning the fitness to an individual, and the process may take many different forms, such as rank-weighting, tournament selection, and so on. Designing a good fitness function is a crucial factor in the success (or otherwise) of the GA, because its convergence will depend strongly on the correlation between the fitness and the distance from the hypothetical optimal solution in genotype space (that is, the Hamming distance). This fitness-distance-

correlation was formalised in a number of works by Jones, Forrest *et al.*[24,52–55]

We can appreciate this aspect of the GA for the simple example of maximising a smooth function. Suppose we choose a fitness functions which is simply proportional to the function value. Then, as the population approaches a relative flat maximum, there may be little difference in fitness between individuals which are nevertheless widely separated in coordinate space (which in the case of a continuous function would be directly given by the genotype). In other words there would be little fitness-distance-correlation in the neighbourhood of a solution and convergence would cease. A simple way to avoid this kind of convergence problem is to rank the individuals in the population by their fitness value, and then to simply equate their fitness with their ranking. This is the aforementioned rank-weighting selection procedure, and is the approach that we will adopt in this study. It has the additional advantage that we do not need to store all fitness values separately, but can keep track of the ranking by re-ordering all the  $N_{\text{pop}}$  genotypes within a single  $N_{\text{pop}} \times \ell$  array. Selection to participate in breeding pairs is then carried out with a linearly distributed likelihood (known generally as roulette wheel selection) that simply depends on an individual's ranking. In the present case an individual at rank  $k$  in the population is selected for breeding with a probability that depends linearly on its ranking as

$$P_k = \frac{2}{(1 + \alpha)N_{\text{pop}}} \left( 1 + \frac{N_{\text{pop}} - k}{N_{\text{pop}} - 1} (\alpha - 1) \right), \quad (2.3)$$

where  $\alpha$  determines how much more frequently the fittest individual is chosen over the least fit. In other words, the probabilities  $P_{N_{\text{pop}}}$  and  $P_1$  for selecting the fittest and least fittest individual are related by

$$P_1 = \frac{2\alpha}{(1 + \alpha)N_{\text{pop}}} = \alpha P_{N_{\text{pop}}}. \quad (2.4)$$

Typically, an optimal configuration is found when the fittest individual is selected for breeding a few times more than the least fit, so that a good choice would be  $\alpha = 2$  or  $3$ . But note that it is important that less fit individuals are also able to breed.

**Breeding/Cross-over:** At this stage we have  $N_{\text{pop}}$  breeding pairs, with the median individual having been selected for breeding twice (that is,  $N_{\text{pop}} P_{N_{\text{pop}}/2} = 1 + \mathcal{O}(1/N_{\text{pop}})$ ). A new population of individuals is then formed by splicing together the genotypes of the two individuals in each breeding pair. Again there are many different ways to do this. The canonical choice is  $N$ -point cross-over, namely one cuts the genotypes of a breeding pair at the same  $N$  random positions along their lengths and swaps the cut sections. For robustness, the publicly available codes often use sophisticated combinations (for example [26, 31, 56, 57] uses both one- and two-point cross-over in roughly equal proportions) to reduce end-point biasing. In the present study this effect was not found to be significant, and a simple one-point cross over was found to work sufficiently well.

**Mutation:** With only the two previous elements, one would already observe clustering of the population around good solutions over generations. However, the power of GAs comes from the third element which is *mutation*. Once a new generation is

formed, a small fraction (usually around a percent) of the alleles are flipped at random. This prevents stagnation in the population, where the entire population clusters around a local maximum in the fitness, when better solutions are available globally. For example, it removes the possibility alluded to below Equation (2.2) that a particular allele could end up being blocked with either all 0's or all 1's running throughout the entire population.

It is important to understand that mutation is not just an improvement to the convergence, but is absolutely integral to the entire process. Depending on the problem and the structure of the fitness landscape, it causes a dramatic increase in the overall rate of convergence. This can be seen in practice by optimising the mutation rate as demonstrated in Ref. [7]. It is worth mentioning one possible innovation that could be considered, but which we did not use in this study, namely so-called *creep mutation*. This is effectively a small mutation in the *phenotype* which overcomes so-called Hamming walls, which occur when the population is close to an optimum solution in terms of phenotype, but far away in terms of Hamming distance. An example would be the binary number 1000 and 0111 which are numerically close but have four different digits. This can again be thought of as a way to overcome breakdowns in the fitness-distance-correlation, and could be thought of as *assisted mutation*.

**Elitism:** Finally to create the new generation in a way that guarantees a monotonically increasing maximum fitness, we invoke *elitism*. This means the fittest individual from the previous generation is copied into the new one and replaces the least fit new individual.

The process is then repeated over many generations, and is terminated and reset when each “perfect model” is found. Note that there are many other practical elements that could be considered, such as fitness “crowding penalties”, and “niching”, that we do not discuss (or require). They are covered in the literature (see Refs. [25, 27]). The overall procedure that we use in this study is summarised in the following pseudo-code (Algorithm 1).

## 2.1. Discussion of Efficacy

Before moving on to the details of heterotic string models with monad bundles, it is worth pausing to consider why GAs are thought to work. With the benefit of several intervening decades of development in heuristic search methods, we can see that GAs contain an interesting mix of elements that can be found in other methods. For example the fitness function is arguably just a cost-function by another name. In addition, the general idea of fitness-distance-correlation in Jones, Forrest *et al.*[24,52–55] clearly has some parallels with Gradient-Descent or Nelder-Mead search techniques. However, one can also see that this correlation can only be a part of the whole story and that GAs must confer their own advantages: indeed a very strong correlation would render the genotype representation superfluous because the dynamics of the flow could be mapped directly to the phenotype. Some power of GAs clearly lies then in the fact that small changes in the genotype can result in huge changes in the phenotype, so that the GA samples the whole space before it begins to converge. Holland's schema theory idealised this behaviour in terms of hidden building blocks within the genotype. A schema is a representation of some crucial set of digits that confers some

Algorithm 1 Pseudocode for GA

---

**Result:** ‘Perfect’ Standard Model built on Calabi-Yau manifold

```

for population do
  | Initialize genotype
end
while generations < gen-number do
  for population do
    | Find phenotype (model properties)
    | Rank individuals by fitness
    | Elitism: Copy fittest individual
  end
  for new-population do
    | Rank-weighted roulette wheel selection of breeding pair from population
    | One-point crossover to create new individuals
    | Mutate new individuals
  end
  Elitism: Paste previous top-ranked individual over bottom-ranked
  population = new-population
end

```

---

favourable characteristic, and in this instance might look something like  $S = 1 *** 00 * .$  This example has 3 entries that we are interested in (hence we say it is order 3) and 4 entries that we do not care about, which are labelled with a wildcard \*. The probabilistic arguments of Holland (summarised in [7]) then suggest that schemata of this kind are important because selection favours the propagation of shorter strings of data: small subsections of the genome that confer fitness dominate first and, once they are shared by the majority of the population, crossover does not affect them. As a result, so the theory goes, if the schema  $S$  confers fitness on an individual, then it will grow exponentially within the population until an equilibrium is reached where the attraction to the solution balances against the repulsion due to the mutation.

Thus the overall behaviour suggested by schema theory is that a GA incorporates and balances competing forces. Selection and breeding tends to produce convergence around local maxima in the fitness landscape, drawing the population in over generations. On the other hand the effect of mutation is to push the population away from local maxima (on average), so that as a whole it can explore the entire parameter space. The power of GAs is then that they are sensitive to the entire landscape, but simultaneously can respond to and converge on interesting regions.

The schema argument fell temporarily out of favour in the 1990s, for reasons which are summarised in Ref. [30], mainly on the grounds that it does not consider the constructive effects of cross-over, and that it assumes a constant contribution to the fitness of a given schema. In reality, the population is not static, so that changes in the rest of the genotype tend to alter the relative fitness of the schema after the first few generations in a run. Despite this objection, the idea was rehabilitated by Poli *et al* in a suite of papers (see Ref. [30] for a comprehensive review) which developed so-called *exact* schema based models, and showed their relation to Markov chains. Simultaneously there

have been several other approaches to understanding GAs, for example statistical mechanical approaches<sup>[58]</sup> or stochastic differential equations,<sup>[59]</sup> which each seem to have their own drawback. In the former approach, one assigns the fitness function with a Boltzmann weighting, while the latter approach is restricted to simple cases such as the so-called onemax problem which is equivalent to solving the one-dimensional Ising model. In these cases it is likely that the set-up of the configuration influences the results so that the conclusions are not general but really only applicable to the very specific configurations used to frame the argument. Despite these limitations, the schema theory remains as one of the best interpretations of the behaviour of the GA.

### 3. Heterotic Models with Monad Bundles

Having presented the main features of GAs, we now turn to a concise discussion of the geometric ingredients entering the construction of  $E_8 \times E_8$  heterotic string compactifications on Calabi-Yau three-folds with monad bundles. More details can, for example, be found in Refs. [10, 60–65]. In particular, we will specify the degree of similarity to the properties of the standard model, that qualifies a particular heterotic model to be designated as “perfect”. From the perspective of the GA this amounts to working out the phenotype.

#### 3.1. Heterotic CY Models

As already alluded to, heterotic  $E_8 \times E_8$  compactifications are based on the data  $(X, V, \tilde{V}, C)$ , where  $X$  is a Calabi-Yau three-fold,  $V$  and  $\tilde{V}$  are slope poly-stable vector bundles over  $X$  whose structure groups embed into  $E_8$  and  $C$  is an effective curve class on  $X$ . Poly-stability is the mathematical condition to ensure the bundles

preserve supersymmetry and it is, in general, difficult and time-consuming to check explicitly. For the purpose of our algorithmic approach we will only check a necessary condition known as Hoppe's criterion. It says that a poly-stable bundle  $V$  over  $X$  must necessarily satisfy

$$h^0(X, \wedge^r V) = 0, \quad \text{for all } r = 1, 2, \dots, \text{rk}(V) - 1. \quad (3.1)$$

Another constraint on the bundle  $V$  arises from the fact that only special unitary groups can be embedded into  $E_8$ . This means that we require

$$c_1(V) = c_1(\tilde{V}) = 0. \quad (3.2)$$

**Anomaly cancellation:** The sectors of the four-dimensional theory which originate from the bundle  $\tilde{V}$  and the five branes wrapping a curve with class  $C$  are hidden sectors, in the sense that they couple to the observable sector associated to  $V$  only gravitationally. Since we are interested in the observable particle physics, these sectors will not be constructed explicitly. However, for a consistent model, we need to satisfy the anomaly cancellation condition

$$c_2(TX) + \text{ch}_2(V) + \text{ch}_2(\tilde{V}) = C \quad (3.3)$$

which relates the various sectors of the theory. This is done by demanding that

$$c_2(TX) + \text{ch}_2(V) \in \text{Mori cone of } X \quad (3.4)$$

which guarantees the existence of a supersymmetric completion of the model with a trivial hidden bundle  $\tilde{V}$  and five-branes wrapping a curve with class  $C = c_2(TX) + \text{ch}_2(V)$ .

**Spectrum:** For the purpose of this paper, we will consider "observable" bundles  $V$  with structure group  $SU(4)$  which breaks one of the  $E_8$  symmetries to an  $SO(10)$  GUT group. The  $SO(10)$  multiplets arise from the decomposition of the 248 adjoint of  $E_8$  under the maximal subgroup  $SU(4) \times SO(10)$ , which reads

$$248_{E_8} \rightarrow \begin{matrix} (1, 45) \\ \text{gauge} \\ \text{bosons} \end{matrix} \oplus \begin{matrix} (4, 16) \\ \text{families} \\ n_{16} = h^1(V) \end{matrix} \oplus \begin{matrix} (\bar{4}, \bar{16}) \\ \text{anti-families} \\ n_{\bar{16}} = h^1(V^*) \end{matrix} \oplus \begin{matrix} (6, 10) \\ \text{Higgs} \\ n_{10} = h^1(\Lambda^2 V) \end{matrix} \oplus \begin{matrix} (15, 1) \\ \text{bundle} \\ \text{moduli} \\ n_1 = h^1(V \otimes V^*) \end{matrix}$$

The number of each multiplet is given by the first cohomology of  $V$ ,  $V^*$  and its various products, as indicated in the last row above. Of particular importance is the chiral asymmetry of families which can be expressed in terms of the index of  $V$  as

$$n_{16} - n_{\bar{16}} = -\text{ind}(V) = h^1(X, V) - h^2(X, V). \quad (3.5)$$

The last equality in the above formula assumes that  $h^0(V) = h^3(V) = 0$  which follows from the poly-stability of  $V$ , via Hoppe's criterion (3.1).

**Equivariance:** The  $SO(10)$  GUT symmetry needs to be broken to the standard model group and this is done by including a discrete Wilson line into the construction, which is possible when

the manifold is non-simply connected. However, most of the typical constructions lead to Calabi-Yau three-folds that are simply connected. On the other hand, if a freely-acting symmetry  $\Gamma$  on  $X$  exists, this can be used to produce a quotient  $\hat{X} = X/\Gamma$  with non-trivial fundamental group. The Wilson line is then included on the resulting quotient manifold  $\hat{X} = X/\Gamma$ . In order to end up with three chiral families after performing this quotient, the upstairs model on  $X$  requires a chiral asymmetry of

$$n_{16} - n_{\bar{16}} = -\text{ind}(V) \stackrel{!}{=} 3|\Gamma|, \quad (3.6)$$

where  $|\Gamma|$  is the order of the discrete group  $\Gamma$ . For this construction to make sense, the upstairs bundle  $V \rightarrow X$  needs to descend to a bundle  $\hat{V} \rightarrow \hat{X}$  on the quotient CY and this is the case iff  $V$  admits a  $\Gamma$ -equivariant structure. This is typically a non-trivial constraint which will be discussed in more detail below, when we introduce monad bundles.

### 3.2. Calabi-Yau Three-Folds

To realise the above construction we require, in a first instance, explicit CY manifolds. It turns out that, for group-theoretical reasons, the Wilson-line breaking of  $SO(10)$  to the Standard Model requires a discrete group  $\Gamma$  which is at least  $\mathbb{Z}_3 \times \mathbb{Z}_3$ . While the number of known CY manifolds is large, only a handful of known examples have symmetry groups this large,<sup>[10,66–68]</sup> so the choice of manifold is, in fact, rather constrained<sup>1</sup>. For the purpose of this paper, we will focus on two of these manifolds, both realised as complete intersections in products of projective spaces (CICYs) and represented by the following configuration matrices

$$\left[ \begin{array}{c|c} \mathbb{P}^2 & 3 \end{array} \right]_{-162}^{2,83}, \quad \left[ \begin{array}{c|ccc} \mathbb{P}^2 & 1 & 1 & 1 \\ \mathbb{P}^2 & 1 & 1 & 1 \\ \mathbb{P}^2 & 1 & 1 & 1 \end{array} \right]_{-90}^{3,48}. \quad (3.7)$$

These manifolds are also known as the bi-cubic CY and the triple tri-linear CY, respectively. They are embedded in the ambient spaces  $\mathcal{A} = (\mathbb{P}^2)^{\times 2}$  for the bi-cubic and  $\mathcal{A} = (\mathbb{P}^2)^{\times 3}$  for the triple tri-linear. The manifold itself is cut out by the common zero locus of homogeneous polynomials whose multi-degrees correspond to the columns of the above matrices. The superscripts provide the Hodge numbers  $h^{1,1}(X)$ ,  $h^{2,1}(X)$  and the subscript the Euler number,  $\eta(X)$ , of the manifold. Both manifolds have a freely-acting symmetry  $\Gamma = \mathbb{Z}_3 \times \mathbb{Z}_3$ , as required for the  $SO(10)$  GUT breaking. We will also need the second Chern class  $c_{2i}(TX)$  of the

<sup>1</sup> This changes for models based on an  $SU(5)$  GUT group whose breaking only requires a  $\mathbb{Z}_2$  symmetry.

tangent bundles and the triple-intersection numbers  $d_{ijk}$ . Relative to the obvious basis  $(J_1, \dots, J_h)$  of  $H^2(X)$ , obtained by restricting the Kähler forms of the projective ambient space factors to  $X$ , and the associated dual basis of  $H^4(X)$ , they are given by

$$\begin{aligned} \text{bi-cubic} \quad c_{2i}(TX) &= (36, 36) & d_{122} &= d_{112} = 3 \\ \text{triple tri-linear} \quad c_{2i}(TX) &= (36, 36, 36) & d_{ijj} &= 3 \quad \forall i \neq j, \\ & & d_{123} &= 6. \end{aligned} \quad (3.8)$$

### 3.3. Monad Bundles

Explicit bundle construction frequently rely on holomorphic line bundles as basic building blocks. Holomorphic line bundles are classified by their first Chern class. Relative to an integral basis  $(J_1, \dots, J_h)$  of  $H^2(X)$ , the first Chern class of a line bundle  $L \rightarrow X$  can be represented by an integer vector  $\mathbf{k} \in \mathbb{Z}^h$ , such that  $c_1(L) = k^i J_i$ , hence the notation  $L = \mathcal{O}_X(\mathbf{k})$ .

Monad bundles are defined by a short exact sequence

$$0 \rightarrow V \rightarrow B \xrightarrow{f} C \rightarrow 0, \quad B = \bigoplus_{i=1}^{r_B} \mathcal{O}_X(\mathbf{b}_i), \quad C = \bigoplus_{a=1}^{r_C} \mathcal{O}_X(\mathbf{c}_a), \quad (3.9)$$

where, for our purposes,  $B$  and  $C$  are each taken to be line bundle sums. Thus such monad bundles are specified by a pair  $(B, C)$  of line bundle sums which can be identified with an integer matrix  $(\mathbf{b}_1, \dots, \mathbf{b}_{r_B}, \mathbf{c}_1, \dots, \mathbf{c}_{r_C})$  of size  $h \times (r_B + r_C)$ . The bundle  $V$  is isomorphic to the kernel,  $\text{Ker}(f)$  of the monad map  $f$  which can be thought of as a  $r_C \times r_B$  matrix with polynomial entries  $f_{ai} \in \Gamma(\mathcal{O}_X(\mathbf{c}_a - \mathbf{b}_i))$ .

A given choice of  $(B, C)$  might lead to a monad sheaf, rather than a monad bundle. This happens when the degeneracy locus of  $f$ , that is, the locus on  $X$  where the rank of  $f$  is less than maximal, is non-trivial. Since we are considering smooth, geometrical models we should avoid sheafs, so we have to check “bundleness” of  $V$  by computing the dimension

$$d_{\text{deg}} = \text{dimension of degeneracy locus of } f. \quad (3.10)$$

Typically, the matrix  $f$  exhibits a pattern of zero and non-zero entries, depending on whether  $\Gamma(\mathcal{O}_X(\mathbf{c}_a - \mathbf{b}_i))$  is trivial or non-trivial, and for otherwise generic choices, the dimension  $d_{\text{deg}}$  can be determined from this pattern. The model is acceptable iff  $d_{\text{deg}} = -1$ .

A monad map with too many zero entries might also lead to a split bundle, that is, to a bundle  $V$  whose structure group is a proper sub-group of  $SU(4)$ , for example  $S(U(2) \times U(2))$ . These split bundles lead to a GUT group larger than  $SO(10)$  and we do not consider this possibility here. To this end, we introduce the number

$$n_{\text{split}} = \text{number of splits of the bundle } V \quad (3.11)$$

which can be determined from the pattern of the monad map. It is also useful to introduce the number

$$n_{\text{trivial}} = \text{number of trivial line bundles } \mathcal{O}_X \text{ in } (B, C). \quad (3.12)$$

As mentioned earlier,  $\Gamma$ -equivariance of  $V$  is a non-trivial constraint but a full check based on presently available methods is not compatible with an efficient computational realisation. Instead, we will perform a strong necessary check for  $B$  and  $C$  to possess a  $\Gamma$ -equivariant structure. This is done by checking that each unique line bundle  $L$  in  $B$  or  $C$  has an index divisible by  $|\Gamma|$ . If a line bundle  $L$  appears with multiplicity  $m$  in either  $B$  or  $C$ , we require that

$$\text{ind}(L^{\oplus m}) \text{ is divisible by } |\Gamma|. \quad (3.13)$$

To check the other constrains we require the following formulae for the Chern classes of a monad bundle:

$$\begin{aligned} \text{rk}(V) &= \text{rk}(B) - \text{rk}(C) \stackrel{!}{=} 4 \\ c_1^k(V) &= c_1^k(B) - c_1^k(C) = \sum_{i=1}^{r_B} b_i^k - \sum_{a=1}^{r_C} c_a^k \stackrel{!}{=} 0 \\ c_{2k}(V) &= \text{ch}_{2k}(C) - \text{ch}_{2k}(B) = \frac{1}{2} d_{klm} \left( \sum_{a=1}^{r_C} c_a^l c_a^m - \sum_{i=1}^{r_B} b_i^l b_i^m \right) \\ &\stackrel{!}{\leq} c_{2k}(TX) \\ \text{ind}(V) &= \sum_{q=0}^3 (-1)^q h^q(X, V) = \frac{1}{2} c_3(V) = \text{ch}_3(B) - \text{ch}_3(C) \\ &= \frac{1}{6} d_{klm} \left( \sum_{i=1}^{r_B} b_i^k b_i^l b_i^m - \sum_{a=1}^{r_C} c_a^k c_a^l c_a^m \right) \stackrel{!}{=} -3|\Gamma|. \end{aligned} \quad (3.14)$$

The conditions appearing on the right-hand sides of these equations originate from the requirement that  $V$  has an  $SU(4)$  structure group, from the anomaly cancellation condition (3.4) and from the constraint (3.6) on the chiral asymmetry.

Calculating cohomology for monad bundles requires the long exact sequence in cohomology associated to the monad sequence (3.9). Such computations can require ranks of maps and can be time-consuming. For this reason, checking Hoppe’s criterion for poly-stability in full is not feasible while running a GA. Instead, we will use that Hoppe’s criterion is violated and, hence, that  $V$  is non-supersymmetric if

$$h^0(B) - h^0(C) > 0 \quad \text{or} \quad h^0(B^*) - h^0(C^*) > 0. \quad (3.15)$$

Fortunately, analytic formulae for zeroth line bundle cohomology exist for both manifolds under considerations,<sup>[10]</sup> so the above expressions can be checked efficiently.

## 4. Monad Bundles and Genetic Algorithms

### 4.1. The Environment

Let us now outline the “environment” that will form the basis of our search. We will consider models consisting of monad bundles (3.9), specified by matrices  $(B, C)$ , on a fixed CY three-fold  $X$ , in practice one of the two manifolds in Equation (3.7). The integer entries in  $(B, C)$  are not, a priori, bounded but will be restricted as

$$b_{\min} \leq b_i^k \leq b_{\max}, \quad c_{\min} \leq c_a^k \leq c_{\max}, \quad (4.1)$$

so that the environment becomes finite. Given that we are aiming to match relatively small integers, such as the required index, this does not seem to be a serious limitation. In fact, model building experience suggests<sup>[6,10,69,70]</sup> that viable models typically arise for relatively small integer entries. The first Chern class constraint,  $c_1(V) = 0$ , can be easily solved (see Equation (3.14)), for example by fixing the last line bundle in  $C$  in terms of the others. For this reason, we constrain the environment to pairs  $(B, C)$  that satisfy  $c_1(B) = c_1(C)$ . If we allowed, say, 10 values per entry, the number of states in the environment is of order

$$10^{h(r_B+r_C-1)}. \quad (4.2)$$

Since  $r_B + r_C - 1 \geq 5$  and we consider  $h = h^{1,1}(X) \in \{2, 3\}$  this is quite sizeable and not suited for systematic scanning. Model building experience also shows<sup>[6,10,69,70]</sup> that the fraction of viable models within this environment is so small that a purely random search could not be successful.

The environment has a large degeneracy, governed by the symmetry

$$H \times S_{r_B} \times S_{r_C}, \quad (4.3)$$

where the first factor is  $H = S_2$  for the bi-cubic CY and  $H = S_3$  for the triple tri-linear CY and arises due to the symmetries of the configuration matrices; it corresponds to permutations of the rows of the matrix  $(B, C)$ . The permutation groups  $S_{r_B}$  and  $S_{r_C}$  arise from permutations of the line bundles (columns) in  $B$  and  $C$  which of course do not affect the resulting monad bundle  $V$ . For the purpose of applying GAs, we have kept this degeneracy, as the ordering required for its elimination would significantly complicate the algorithm. Indeed the very notion of ordering by genotype, which would have to be performed even after each mutation, is somewhat at odds with the principle of GAs although it could in principle be included. However, redundancy in the models will be ultimately be eliminated from the list of perfect models found by the algorithm.

For the purpose of implementing the GA we require an intrinsic value function  $\nu(B, C)$  on the environment which measures any deviation from the required properties and, in the context of GAs, forms the basis of the fitness of a given individual. The various contributions to this value function are listed in **Table 1**. A viable or perfect model  $(B, C)$  is defined to be a model with intrinsic value  $\nu(B, C) = 0$ . Given the definition of the value function in Table 1, such a state allows for a supersymmetric, anomaly-free completion, it is based on an  $SO(10)$  GUT symmetry, has the correct chiral asymmetry of families and it passes the checks for bundle equivariance and poly-stability discussed earlier. Of course such perfect models might then still fail at a fairly elementary level, for example due to the presence of undesirable  $\overline{16}$ -multiplets, a failure of stability or a failure of equivariance. Checking these properties requires cohomology calculations which, with the available methods in commutative algebra, are too time-consuming to be carried out during the GA evolution. However, experience from “by-hand” model building<sup>[6,69,70]</sup> and from RL<sup>[10]</sup> shows that a significant fraction of the perfect models also satisfy the more stringent coho-

**Table 1.** Contributions to the intrinsic value for the monad environment. The intrinsic value  $\nu(B, C)$  is the sum of all eight terms. The normalisation factors involve  $M = \max(b_{\max}, c_{\max})$  and  $h = h^{1,1}(X)$ .

Property	Term in $\nu(B, C)$	Comment
index match	$-\frac{2 \text{ind}(V)+3 \Gamma }{hM^3}$	$\text{ind}(V)$ computed from Equation (3.14)
anomaly	$\sum_{i=1}^h \min(c_{2i}(TX) - c_{2i}(V), 0)$	$c_{2i}(V)$ computed from Equation (3.14)
bundleless	$-(d_{\text{deg}} + 1)$	$d_{\text{deg}}$ from Equation (3.10)
split bundle	$-n_{\text{split}}$	$n_{\text{splits}}$ as in Equation (3.11)
equivariance	$-\sum_{L \subset B, C} \text{mod}(\text{ind}(L^{\otimes m}),  \Gamma )$	$L$ and $m$ is in Equation (3.13)
trivial bundle	$-n_{\text{trivial}}$	$n_{\text{trivial}}$ as in Equation (3.12)
stability $V$	$-\frac{\max(0, h^0(X, B) - h^0(X, C))}{hM^3}$	test from Equation (3.15)
stability $V^*$	$-\frac{\max(0, h^0(X, B^*) - h^0(X, C^*))}{hM^3}$	test from Equation (3.15)

mological constraints. In other words, while some of the conditions in Table 1 are only necessary, they are still sufficiently strong<sup>2</sup>.

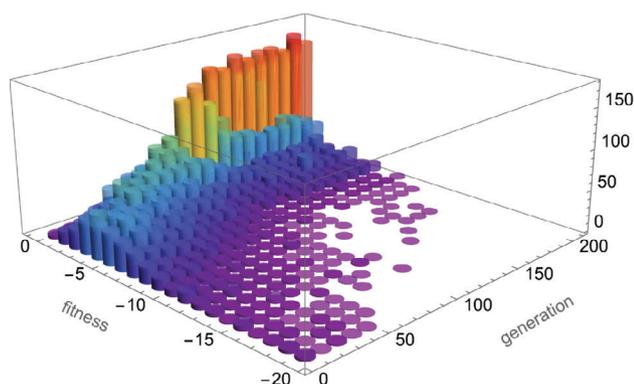
The monad environment we have just described was also used in the context of RL,<sup>[10]</sup> where the value function in Table 1 was used to determine the reward. It was shown in Ref. [10] that RL is extremely successful on this environment, with trained policy networks efficiently leading to perfect models for 100% of episodes, and many, previously unknown, models being discovered. Using the same environment allows a direct comparison between RL and GAs, which we carry out in the next section. In summary then, our environment is the same as that used for RL in Ref. [10], which has been realised as a MATHEMATICA<sup>[71]</sup> package.

## 4.2. Genetic Algorithm

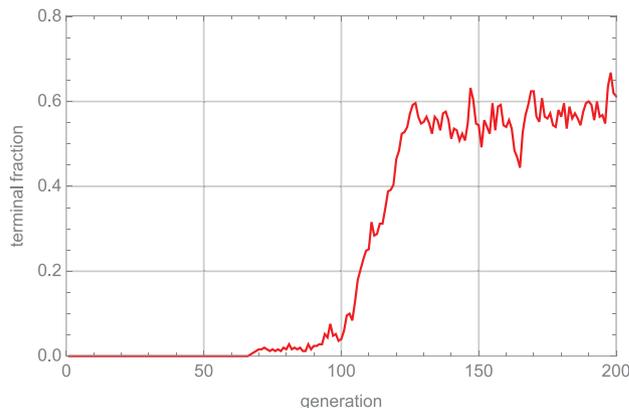
To realise the GA we choose the range of entries such that  $b_{\max} - b_{\min} = 2^{n_B} - 1$  and  $c_{\max} - c_{\min} = 2^{n_C} - 1$ , representing every integer in  $B$  and  $C$  in a binary encoding, with  $n_B$  and, respectively,  $n_C$  bits, resulting in a bit sequence of total length  $\ell = h(r_B n_B + (r_C - 1)n_C)$ . The typical size of a population consists of  $N_{\text{pop}} \simeq 100 - 300$  states, each represented by such a binary encoding of the matrix  $(B, C)$ . The population is then evolved by crossing and mutation, as discussed in Section 2.

With the fitness function specified by the intrinsic value  $\nu(B, C)$ , we found that rank-weighting selection, elitism and a simple one-point cross breeding work well. The GA was implemented in two ways, as a MATHEMATICA<sup>[71]</sup> and as a Python package, in both cases being coupled to the MATHEMATICA package realising the environment. The two realisations lead to similar results.

<sup>2</sup> It is conceivable that analytical formulae for monad cohomology can be found, in analogy with the analytical formulae known for line bundle cohomology.<sup>[13,16–18]</sup> Such formulae would facilitate setting up a more advanced environment which checks the full spectrum and a sufficient condition for stability.



(a) Fitness histogram: number of individuals as a function of generation and fitness.



(b) Fraction of perfect models vs generation.

**Figure 1.** Performance measures for a typical GA initialisation on the bicubic.

## 5. Monad Bundles on the Bicubic Manifold

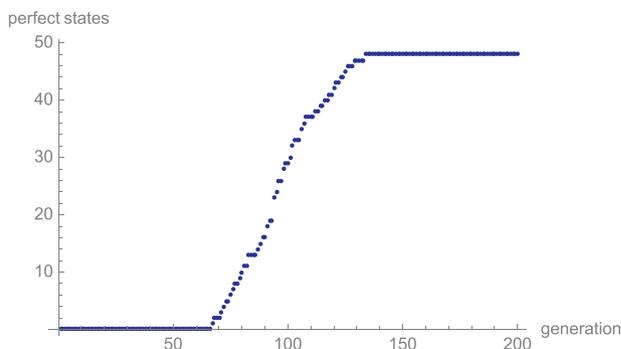
### 5.1. A Typical GA Run on the Bicubic

We first consider monad bundles on the bicubic manifold with  $\text{rk}(B) = 6$  and  $\text{rk}(C) = 2$ . We choose  $b_{\min} = -3$ ,  $c_{\min} = 0$  and  $b_{\max} = 4$ ,  $c_{\max} = 7$ , such that each integer entry in the matrix  $(B, C)$  can be represented by a 3-bit binary sequence counting up from the fiducial points,  $b_{\min}$  and  $c_{\min}$ . For instance, an entry of  $-3$  in  $B$  will be represented by the sequence  $(0,0,0)$ , while an entry of  $4$  in  $B$  corresponds to  $(1,1,1)$ . Similarly, an entry of  $0$  in  $C$  is represented as  $(0,0,0)$ , while an entry of  $7$  in  $C$  corresponds to  $(1,1,1)$ .

Working with integers that can be encoded on 3 bits is convenient for the purpose of comparing the GA and RL techniques. However, it is possible and in fact easy to further increase the range of line bundle integers, as the corresponding augmentation in the computational time is not significant.

As mentioned above, the environment has a large degeneracy. Equivalent models arise from permuting the two  $\mathbb{P}^2$ -factors in the bicubic embedding, as well as from permuting the line bundles in  $B$  and  $C$ . This amounts to a group of order  $2! \cdot 6! \cdot 2! = 2800$ . Of course, this does not necessarily imply that every perfect model can be found in 2800 different sites, since some permutations may leave particular matrices  $(B, C)$  unchanged, but it nevertheless gives some idea of the degree of degeneracy present in the environment.

It is useful to illustrate the performance of the GA graphically. We set the size of the population to  $N_{\text{pop}} = 250$  randomly initialised states and let the GA run for 200 generations with a mutation rate of 0.004 and the parameter  $\alpha$  from Equation (2.3) set to  $\alpha = 3$ . The time taken by such a run is relatively small (<100 seconds on a laptop). **Figure 1** shows a typical evolution of the population. The histogram on the left shows how the population progresses towards greater fitness (intrinsic value). The plot on the right shows the fraction of perfect models in the population, that is, the fraction of models with a value  $v(B, C) = 0$ . By generation 150 more than half of the population corresponds to perfect models. For this run, a total of  $200 \times 250$  states have been vis-



**Figure 2.** Saturation of the number of perfect states found in a typical GA run on the bicubic.

ited, many of them multiple times. Of these 12,665 correspond to perfect models with 48 being distinct. After eliminating redundancies, 18 non-equivalent models remain. The fraction of perfect models can be easily increased by lowering the mutation rate. However, as one might anticipate from the discussion in the introduction, this also induces a degree of stagnation, with a tendency for the same states to emerge repeatedly.

A few remarks are in order. Firstly, to appreciate the performance of the algorithm, note that the size of the search space is

$$8^{14} \approx 4.4 \times 10^{12}.$$

By comparison, the number of states visited in the above run, namely 50,000, represents only a tiny fraction of the space. However, the GA was capable of finding 48 perfect states, while a random search over millions of states would typically lead to no perfect states at all.

Secondly, as already noted, the GA has a tendency of visiting the same states multiple times. It is interesting to plot the total number of perfect states found after  $n$  generations as a function of  $n$ . For our illustrative run such a plot is shown in **Figure 2**,

which suggests that there is no additional benefit in letting the population evolve beyond a certain generation ( $n \simeq 150$ ).

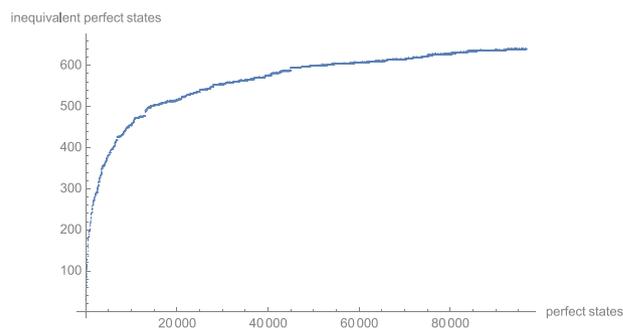
Finally, it is useful to compute the degeneracy of the 18 states that remain after removing redundancies. By performing all the allowed permutations a number of 19,080 states are obtained. This should be compared with the product  $18 \times 2800 = 50,400$  which turns out to be an overestimate by more than a factor of 2. Moreover, what this computation indicates is that a single run of the GA is not enough if the aim was to find all the perfect states available in the environment and that 1,000 further runs, which would take about 1 day on a single machine, would be just about enough to find the other redundant representations of the 18 states found in the first run. Of course, many more new states, not related to the 18 by permutations, would be expected to arise in such a search, which implies that a comprehensive search would require several, possibly tens of thousands of GA runs. With 10,000 runs this would amount to exploring  $\sim 0.01\%$  of the environment.

### 5.2. A Quasi-Comprehensive GA Search

The illustrative results presented above show that GAs can be efficiently used to identify monad bundle models on the bicubic manifold that satisfy the phenomenological requirements discussed in Section 3. A natural question to ask is whether the GA can find *all* (or almost all) perfect models available within the environment. Answering this question is difficult, mainly because the environment is too large to be subjected to a systematic scan which would ensure that no perfect state was missed. The computational time needed for a systematic scan within the environment bounds can be easily estimated to about 200 core years, which although attainable, is by no means something that one would easily invest in.

A partial answer to the question can be achieved by making use of the redundancy present in the environment. The procedure is as follows. We perform 10,000 runs keeping the same settings as above, namely  $N_{\text{pop}} = 250$ , the length of a run equal to 200 generations, a mutation rate of 0.004 and  $\alpha = 3$ . The required computation time for this is about 10 core days<sup>3</sup>. We extract the number of perfect states found after  $n$  runs and then ask how many of these are left after removing redundancies. Now, if the GA is capable of finding almost all the states, then the expectation would be that the redundancy reduced number of perfect states after  $n$  runs would saturate as a function of the total number of perfect states found after  $n$  runs. In other words the GA is only finding versions of models that it has found before. The plot in **Figure 3** confirms this expectation. The search lead to a number of 96,705 perfect states, out of which 639 are inequivalent (not related by a permutation). Performing all possible permutations on the 639 states leads to 481,680 perfect states that must be present in the environment. If allowed more time, the GA would of course be expected to find all of these. However, the saturation plot in **Figure 3** as well as the plots in **Figure 4** suggest that there are not many more inequivalent states to be found. We take this as an indication that the search has already successfully found most of the inequivalent perfect states.

<sup>3</sup> The scan was carried out on the Hydra Computer Cluster in the Dept. of Physics at the University of Oxford.



**Figure 3.** GA scan results. Saturation of the redundancy reduced number of perfect states.

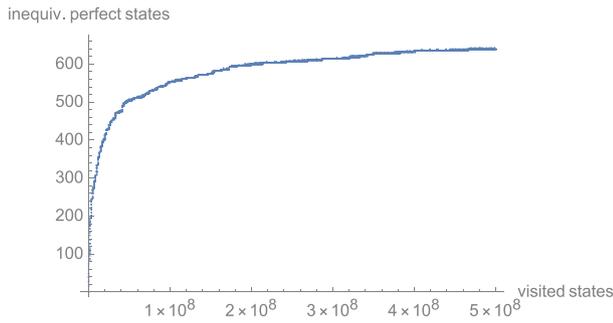
### 5.3. Enlarging the Search Space

So far the minimal and the maximal values of the entries in  $B$  and  $C$  were fixed such that all integers can be encoded on 3 bits. Upgrading to integers that can be encoded on 4 bits leaves the computational time required for a GA run virtually unchanged. Thus enlarging the range of line bundle integers in  $B$  and  $C$  would not pose a problem for the performance of the GA implementation used here. But is there a good reason for enlarging the environment?

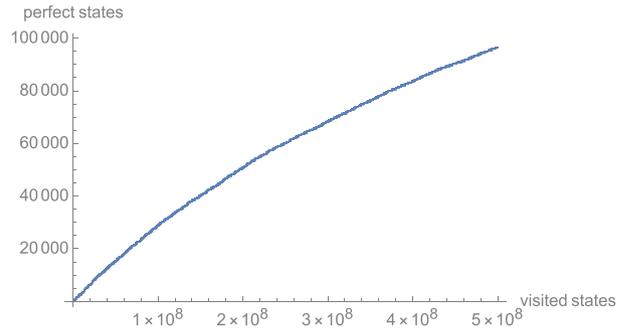
To answer this question we need to see if the perfect models found are well within the boundary of the environment or if a significant fraction of them lies close to the boundary. The bar charts in **Figure 5** show that in fact most of the models lie on the boundary (for instance, 451 out of the 639 inequivalent models include the integer 4 in  $B$ ). This suggests the need for increasing further the entries in  $B$  and  $C$ . However, before embarking on such an enterprise, it is worth looking deeper into the quality of the monad bundles found. The requirement of stability imposes that  $h^0(V) = h^3(V) = 0$ . Moreover, in order to avoid the presence of  $\overline{16}$ -multiplets, the bundle has to satisfy  $h^2(V) = 0$ . These cohomological constraints are too computationally intensive to have included them in the search, however they can be carried out on the set of 639 inequivalent models found on the bicubic manifold. Moreover, many of the models found contain a repeated line bundle in  $B$  and  $C$ . While this is not a problem in itself, such models are equivalent to monad bundles with  $\text{rk}(B) = 5$  and  $\text{rk}(C) = 1$ . We eliminate these from the discussion, which leaves 214 models of genuine  $\text{rk}(B) = 6$  and  $\text{rk}(C) = 2$  type. Out of these, 67 have the right cohomological properties mentioned above. Further stability checks reduce this number to 29 models. The distribution of entries in  $B$  for the remaining 29 models is shown in blue in **Figure 6** and indicates that most of the models have all the line bundle entries in the range  $\{-2, -1, 0, 1, 2\}$ . In view of this, the expectation is that very few, if any, additional models satisfying the additional constraints would be found by further enlarging the size of the environment.

### 5.4. GA Scan Versus RL Scan

In Ref. [10] a similar search for physically viable models on the bicubic manifold with monad bundles was performed using methods of reinforcement learning (RL). The same fitness function  $\nu(B, C)$  as described above was used in that context to

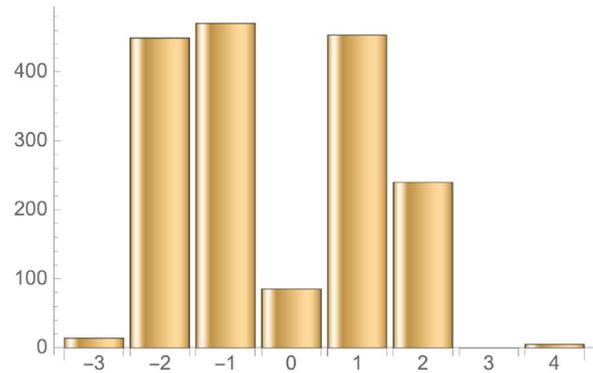
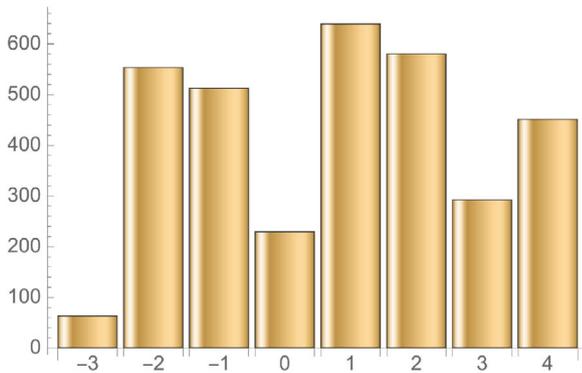


(a) Number of inequivalent perfect states found as a function of the number of states visited.



(b) Number of perfect states found as a function of the number of states visited.

**Figure 4.** GA search results on the bicubic. The number of inequivalent perfect states saturates, while the total number of perfect states found is still far from saturation after 10,000 GA runs.

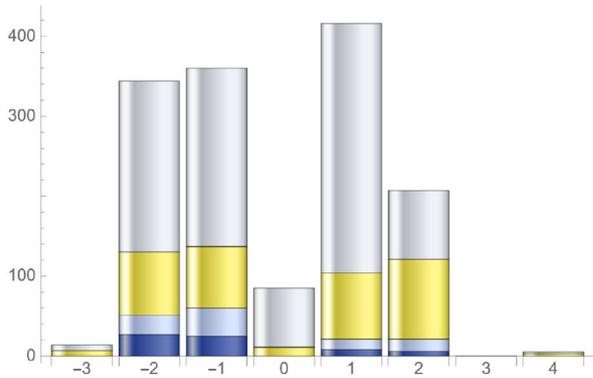
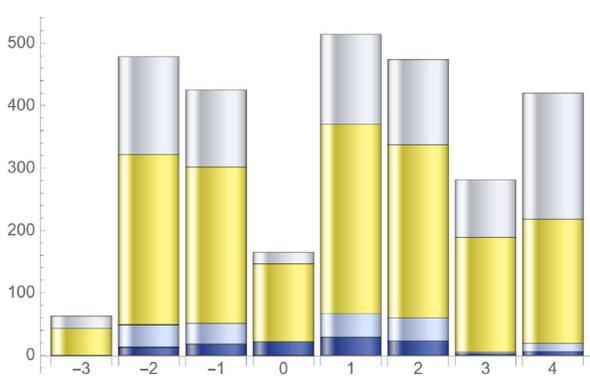


**Figure 5.** Bar charts showing the number of inequivalent perfect states ( $B, C$ ) that have a certain integer present in  $B$  (left chart) and in  $C$  (right chart).

compute the intrinsic value of states ( $B, C$ ). When moving from one state to another, the RL agent receives a reward equal to the difference between the intrinsic values of the two states if the difference was positive or a fixed penalty if the difference was negative. The search is divided into episodes of fixed maximal

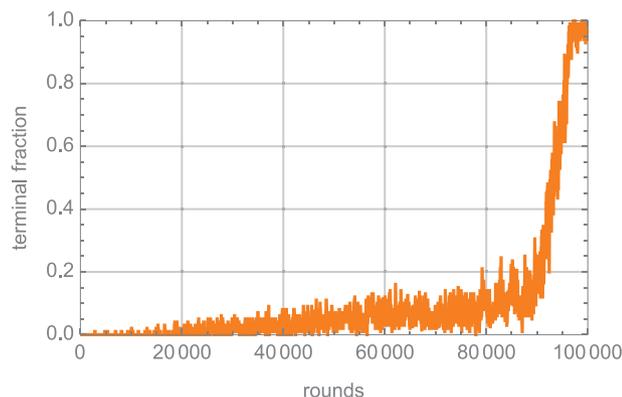
length and along each such episode the return  $G_t$  of a state  $s_t$  is computed as

$$G_t = \sum_{k \geq 0} \gamma^k r_{t+k}, \quad (5.1)$$

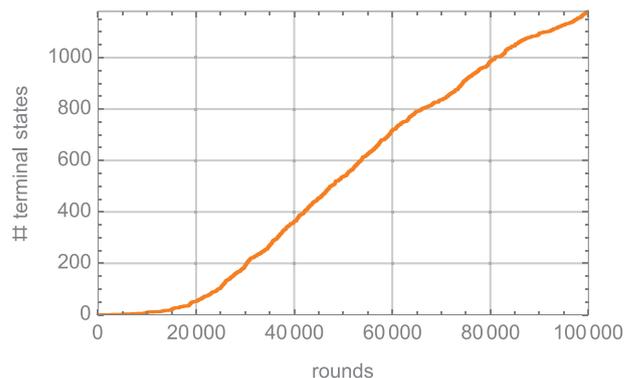


■ models found (inequivalent)  
■ models of genuine type (6,2)  
■ models of genuine type (6,2) satisfying cohomological constraints  
■ stable models of genuine type (6,2) satisfying cohomological constraints

**Figure 6.** Distribution of models across the range of integers in  $B$  (top chart) and  $C$  (bottom chart). The number of models is significantly reduced when additional requirements are imposed. Very few models are left on the boundary of the environment.



**Figure 7.** RL training metrics on the bicubic manifold. After 100,000 rounds every episode ends up in a terminal state.



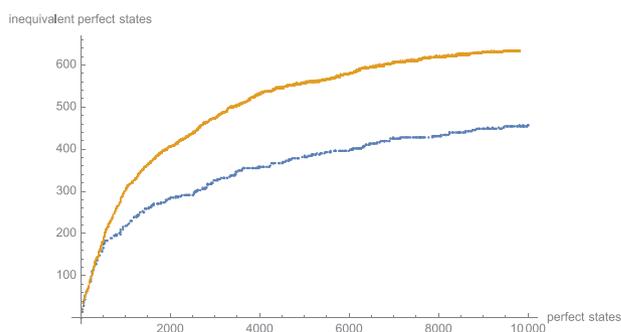
where  $r_i$  is the reward (penalty) associated with the step  $s_i \rightarrow s_{i+1}$  and the discount factor  $\gamma$  is a subunitary number close to 1. The aim of the game is then to devise a strategy of finding a path that would maximise the return of any randomly chosen starting point, by eventually landing on a perfect state (also called a terminal state in the RL context).

More details can be found in Ref. [10]. What is important for the present discussion is that, although the RL and GA methods rely on the same intrinsic value function, their underlying philosophies are very different, which makes for an interesting comparison.

For the RL we use a policy network trained on episodes of maximal length 64. After 100,000 rounds of training, a phase which takes several hours on a laptop, the RL agent is capable of finding terminal states from virtually any random starting point within episodes of average length  $\sim 30$ . The training curves are shown in **Figure 7**.

In order to get a preliminary assessment of the performance of the trained network we let it run for 10,000 episodes which takes about 16 minutes on a single machine. The run produced 2460 terminal states, out of which only 72 are distinct. The repetitions are not evenly distributed over the 72 distinct states. In fact, one of these states accounts for almost half of the terminal states found. Incidentally, this particular state corresponds to a monad bundle that can be reduced to a  $\text{rk}(B) = 5$  and  $\text{rk}(C) = 1$  case. The genuine  $\text{rk}(B) = 6$  and  $\text{rk}(C) = 2$  are found with much smaller frequencies, most of them appearing only once in the dataset. Thus, although the network is capable of finding terminal states, a small number of these states have very large basins of attraction and are found much more often than the others. This is a source of inefficiency for the RL implementation.

We let the trained network search for terminal states from  $1.7 \cdot 10^9$  random initial states. This search, which took 35 core days produced  $\sim 10,000$  perfect states, out of which 643 were inequivalent. By comparison, the computational time for the GA scan was 10 core days and led to  $\sim 100,000$  perfect states, out of which 639 were inequivalent. The saturation curve of the number of inequivalent models as a function of the total number of models found at various stages of the RL search is shown in **Figure 8** together with the corresponding curve for the first 10,000 perfect states found during the GA search. **Figure 9** shows the progression of the number of (inequivalent) models found with the number of states visited.

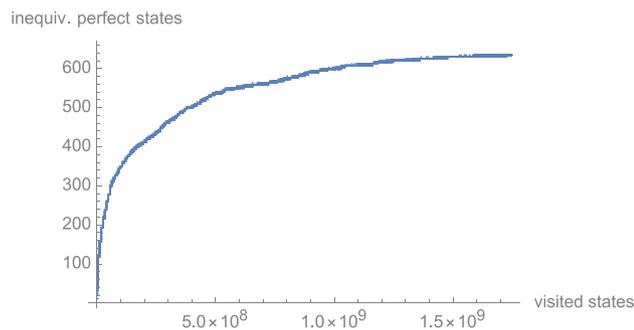


**Figure 8.** Saturation of the redundancy reduced number of perfect states. Orange curve: RL search. Blue curve: GA search. The RL search took 35 core days, while the part of the GA search shown here took only 1 core day.

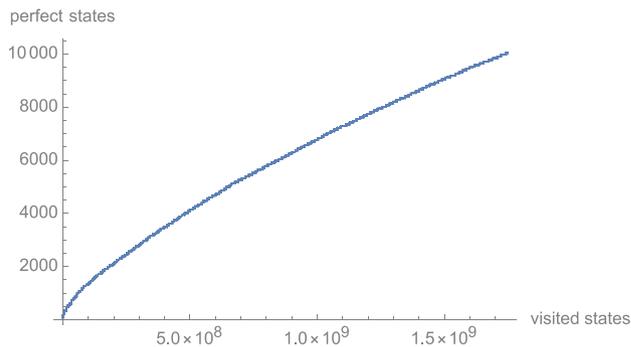
A number of remarks can now be made. Firstly, the GA search is more efficient at finding perfect states than the RL search. Roughly the same number of inequivalent perfect states have been found in the two searches, though the RL search took 3.5 times longer than the GA. Secondly, the GA is more prolific, by more than an order of magnitude compared to the RL search, at finding redundant perfect states. This is very likely a by-product of crossover, which is prone to producing redundant states by permuting various parts of the genome. Note that various techniques such as crowding penalties might be considered to remove this inefficiency.

Finally, there is a substantial overlap between the inequivalent models found in the two searches, with only  $\sim 50$  models in each complement. This brings the total number of inequivalent models to 689. The fact that two such different methods lead to virtually the same dataset of models is another even stronger indication that it has been possible to achieve a high degree of comprehensiveness by scanning only a tiny fraction of the environment.

While both methods produce nearly identical data sets of perfect models after sufficient running time their trajectories during evolution/training appear to be quite different. Using nonlinear visualisation techniques such as Sammon maps we find that the set of perfect models found at an early stage is quite different for the two methods. In particular, as illustrated by **Figure 8**, GAs tend to produce a significantly larger degree of redundancy than RL.

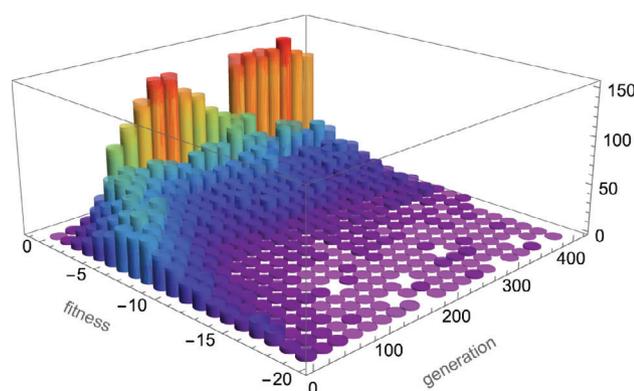


(a) Number of inequivalent perfect states found as a function of the number of states visited.

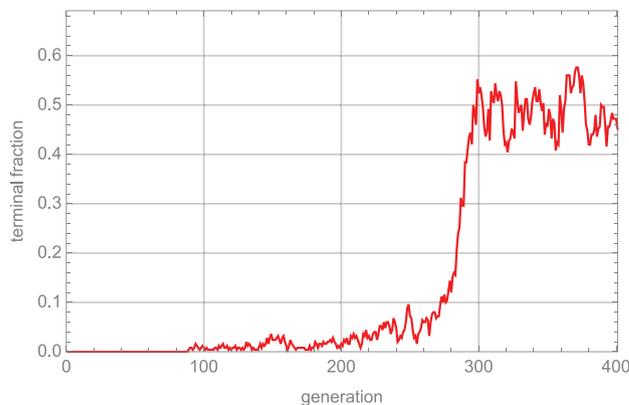


(b) Number of perfect states found as a function of the number of states visited.

**Figure 9.** RL search results on the bicubic. The number of inequivalent perfect states saturates, while the total number of perfect states found is still far from saturation after  $1.7 \cdot 10^9$  RL episodes.



(a) Fitness histogram. Height: number of individuals. Front axis: fitness (intrinsic value). Depth: generation number.



(b) Fraction of models that achieved 'perfection' vs generation.

**Figure 10.** Performance measures for a typical GA initialisation on the triple trilinear.

## 6. Monad Bundles on the Triple Trilinear Manifold

### 6.1. A Typical GA Run on the Triple Trilinear

As in the previous section we consider monad bundles with  $\text{rk}(B) = 6$  and  $\text{rk}(C) = 2$  and set  $b_{\min} = -3$ ,  $c_{\min} = 0$  and  $b_{\max} = 4$ ,  $c_{\max} = 7$ , such that each integer entry in the matrix  $(B, C)$  can be represented by a 3-bit binary sequence. For the triple trilinear manifold the degeneracy of the environment is even larger than in the case of the bicubic, due to the six possible permutations of the three  $\mathbb{P}^2$ -factors in the ambient space. The corresponding group has order 8640.

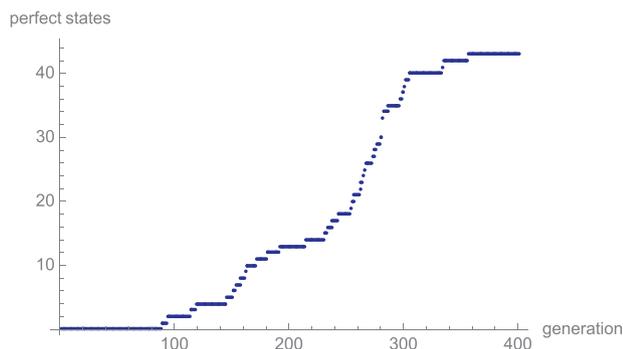
As in the previous case, we illustrate the performance of the GA graphically. We keep the size of the population to  $N_{\text{pop}} = 250$  randomly initialised states. The GA needs to run a little longer until a significant fraction of the population corresponds to perfect states. Evolutionary episodes of 400 generations with a mutation rate of 0.004 and  $\alpha = 4$  seem to work well. The time taken by such a run is relatively small ( $\sim 450$  seconds on a single machine). **Figure 10** shows a typical evolution of a population of models

on the triple trilinear manifold. By generation 300 about half of the population corresponds to perfect models. For this run, a total of  $400 \times 250$  states have been visited, many of them multiple times. Of these 15,377 correspond to perfect models, with 43 being distinct. After eliminating redundancies, 41 non-equivalent models remain.

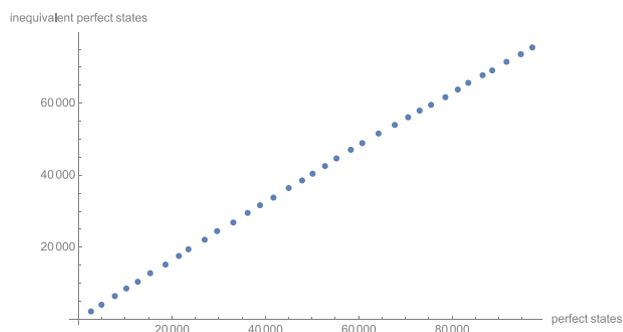
The size of the search space is

$$8^{21} \simeq 10^{19}.$$

which is orders of magnitude larger than the bi-cubic space. The number of states visited during the above run, namely 100,000 states (not necessarily distinct), represents a minuscule fraction of the space. Nevertheless, it is remarkable that 43 perfect states were found, and these perfect states lead to 41 inequivalent models. The total number of perfect states found after  $n$  generations as a function of  $n$  is plotted in **Figure 11**, which suggests that the number of generations, 400, is suitably chosen given the other hyper-parameters. This continued efficacy despite the much larger search space suggests that the difficulty of finding a



**Figure 11.** Saturation of the number of perfect states found in a typical GA run on the triple trilinear CY manifold.



**Figure 12.** GA search results on the triple trilinear manifold. The creation of each point has taken 10 core hours and corresponds to  $10^8$  visited states. The number of inequivalent perfect states has not saturated after  $35 \cdot 10^8$  visited states.

solution is increasing only polynomially for the GA as one might have hoped.

## 6.2. GA Search on the Triple Trilinear

The environment being much larger than in the case of the bicubic manifold, we have not attempted to reach the same degree of comprehensiveness as before. After running for 15 core days, the GA found  $\sim 100,000$  perfect states, three quarters of which are inequivalent. **Figure 12** shows that the number of inequivalent states is far from saturation at the end of the run, suggesting that many more other inequivalent perfect states are still to be found within the environment.

## 7. Conclusion

In this paper we have explored the efficacy of genetic algorithms (GAs) as a search method for probing the landscape of four-dimensional  $\mathcal{N} = 1$  heterotic compactifications. GAs turn out to be very successful for engineering the compactification data and for finding perfect models, that is, models which resemble the Standard Model. The spaces of models scrutinised here were extremely large ( $\sim 10^{12}$  in the case of the bicubic manifold and  $\sim 10^{19}$  for the triple trilinear manifold). In the case of the bicubic manifold a systematic scan, although feasible, would be extremely time consuming, while for the triple trilinear manifold,

a systematic scan is practically unfeasible. Despite this, the GA was able to find tens of models within the first few minutes of running on a single machine. The method is not only efficient, but also highly parallelisable.

After 10 days of running, the GA accomplished a high degree of comprehensiveness in finding all the perfect models available on the bicubic manifold. Indirect evidence in support of this claim was obtained by exploiting the degeneracy of the environment, as well as by comparison of the results of the GA search with those obtained through reinforcement learning (RL). Although the two methods are very different, they lead to datasets of inequivalent models that have an overlap of over 90%. This suggests that the details of the optimisation process are not really essential once the processes begin to saturate, provided that they share the same incentives.

Our comparative analysis of the two search methods revealed that GA is, overall, more efficient by about an order of magnitude in finding perfect states than RL. However, GA is also more prone to finding models that are related by redundant permutations, a feature which is likely related to the crossover mechanism. In principle techniques such as crowding penalties (with the redundancy incorporated) might improve this aspect. An important factor contributing to the lower output of the RL search was the fact that certain perfect states had very large basins of attraction and consequently were found many times, while other perfect states had small basins of attraction and needed many trial episodes to be found.

The combined RL and GA searches on the bicubic manifold led to some 700 inequivalent perfect models. Only a tenth of these correspond to monad bundles of genuine  $(\text{rk}(B), \text{rk}(C)) = (6, 2)$  type with the correct cohomology dimensions. Further stability checks reduce the number of viable models by another factor of 2 – the corresponding monad bundles have been tabulated in Appendix A. Unfortunately these checks had to be performed at the end of the search due to the limitations imposed by cohomology computations. However, it may be the case that a quick method for computing monad bundle cohomology can be developed, for instance in the form of an analytic formula similar to those found for line bundle sums. If possible, this would facilitate the implementation of further checks at the search stage, which would significantly improve the performance of both GAs and RL.

Improving the performance of the search methods is of particular importance for large environments, such as that corresponding to the triple trilinear manifold. In the absence of more stringent notion of perfect model, achieving any degree of comprehensiveness is difficult in such large environments; consequently, partial searches such as the one presented in Section 6.2 may miss many relevant models.

Looking more broadly, one is irresistably drawn to the question of whether the optimisation processes that we have been exploring here might have a bearing on physics in the early Universe, in the sense that a qualitatively similar kind of selection may have happened involving, for instance, topological and small-instanton transitions between different vacua. Of course it is not known what (if any) incentives might favour the physics that we observe in our world, although there have over the years been varyingly speculative suggestions (and we do not need to add to the list). But if this is the case and if the idea of a multiverse is correct, then the results that we observe with GA and RL searches do

at least support the idea that selection mechanisms can successfully operate within the vast search space, and indeed that they do so in a quite convergent way. This suggests the possibility of alternative pictures, quite distinct from that of a randomly populated multiverse, in which the properties of this world would be more common than one would naively expect.

## Appendix

### A. Monad bundles on the bicubic

The table below contains the 28 monad bundles of genuine type  $(r_B, r_C) = (6, 2)$  obtained on the bicubic manifold  $X$ , leading to models with 27 families and no anti-families. For these bundles equivariance with respect to the  $\mathbb{Z}_3 \times \mathbb{Z}_3$  action on the manifold has been checked at the level of index divisibility for the line bundles in  $B$  and  $C$ . A number of necessary stability checks have been carried out. More precisely, we have found all the line bundles with entries in the range  $[-3, 3]$  that inject into  $V$  and have checked that there exists a non-empty locus in the Kähler cone where all these line bundles have a negative slope, thus leaving  $V$  slope poly-stable.

In order to save space we use the notation  $\mathcal{O}(k_1, k_2)$  instead of  $\mathcal{O}_X(k_1, k_2)$ . All the line bundles are to be understood on the bicubic threefold.

$B$	$C$
$\mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(1, -1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(2, -1) \oplus \mathcal{O}(2, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(1, -1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 2)$	$\mathcal{O}(1, 4) \oplus \mathcal{O}(2, 1)$
$\mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0)$	$\mathcal{O}(1, 1) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(2, -1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(1, -2) \oplus \mathcal{O}(1, 2) \oplus \mathcal{O}(2, 1)$	$\mathcal{O}(2, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-2, 1) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 2) \oplus \mathcal{O}(2, 1)$	$\mathcal{O}(2, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-2, 2) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 2) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(2, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 2) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(2, -2) \oplus \mathcal{O}(2, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(0, 2) \oplus \mathcal{O}(0, 2) \oplus \mathcal{O}(0, 2) \oplus \mathcal{O}(1, -2) \oplus \mathcal{O}(1, -1) \oplus \mathcal{O}(1, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 2) \oplus \mathcal{O}(-1, 2) \oplus \mathcal{O}(1, -1) \oplus \mathcal{O}(1, -1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(2, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(2, -1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(2, -1) \oplus \mathcal{O}(2, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 1) \oplus \mathcal{O}(2, 0) \oplus \mathcal{O}(2, 0) \oplus \mathcal{O}(2, 0)$	$\mathcal{O}(1, 1) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 2) \oplus \mathcal{O}(2, -1)$	$\mathcal{O}(1, 4) \oplus \mathcal{O}(2, 1)$
$\mathcal{O}(-2, 2) \oplus \mathcal{O}(-1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 4) \oplus \mathcal{O}(2, -1) \oplus \mathcal{O}(2, -1)$	$\mathcal{O}(1, 5) \oplus \mathcal{O}(2, 1)$
$\mathcal{O}(-2, 1) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 2) \oplus \mathcal{O}(1, 4)$	$\mathcal{O}(1, 5) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 2) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 4)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(1, 5)$
$\mathcal{O}(-1, 2) \oplus \mathcal{O}(-1, 2) \oplus \mathcal{O}(1, -1) \oplus \mathcal{O}(1, -1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 4)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(1, 5)$
$\mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(1, -2) \oplus \mathcal{O}(1, 2) \oplus \mathcal{O}(1, 4)$	$\mathcal{O}(1, 5) \oplus \mathcal{O}(2, 2)$
$\mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(1, -2) \oplus \mathcal{O}(1, 4) \oplus \mathcal{O}(4, 1)$	$\mathcal{O}(1, 5) \oplus \mathcal{O}(5, 1)$
$\mathcal{O}(-1, 2) \oplus \mathcal{O}(-1, 2) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(2, -1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 1)$
$\mathcal{O}(-2, 2) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 2)$	$\mathcal{O}(1, 4) \oplus \mathcal{O}(2, 1)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1) \oplus \mathcal{O}(1, 1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(1, 4)$
$\mathcal{O}(-1, 1) \oplus \mathcal{O}(-1, 2) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 2)$	$\mathcal{O}(1, 1) \oplus \mathcal{O}(1, 4)$
$\mathcal{O}(-2, 1) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 0) \oplus \mathcal{O}(1, 4) \oplus \mathcal{O}(4, 1)$	$\mathcal{O}(1, 5) \oplus \mathcal{O}(5, 1)$
$\mathcal{O}(-1, 2) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(0, 1) \oplus \mathcal{O}(2, -1) \oplus \mathcal{O}(2, -1)$	$\mathcal{O}(1, 2) \oplus \mathcal{O}(2, 1)$

## Acknowledgements

A. C. is supported by a Stephen Hawking Fellowship, EPSRC grant EP/T016280/1, and T. R. H is supported by an STFC studentship. We would like to thank Sven Krippendorf for suggesting GAs as a complementary search method to RL.

## Conflict of Interest

The authors have declared no conflict of interest.

## Keywords

genetic algorithms, heterotic string compactifications, reinforcement learning, SO(10) GUTs

Received: March 1, 2022

Revised: March 1, 2022

Published online:

- [1] M. R. Douglas, *JHEP* **2003**, 05, 046, hep-th/0303194.
- [2] S. Ashok, M. R. Douglas, *JHEP* **2004**, 07, 060, hep-th/0307049.
- [3] W. Taylor, Y.-N. Wang, *JHEP* **2015**, 12, 164, 1511.03209.
- [4] A. Constantin, Y.-H. He, A. Lukas, *Phys. Lett. B* **2019**, 792, 258, 1810.00444.
- [5] M. Cvetič, J. Halverson, L. Lin, M. Liu, J. Tian, *Phys. Rev. Lett.* **2019**, 123, 101601, 1903.00009.
- [6] L. B. Anderson, A. Constantin, J. Gray, A. Lukas, E. Palti, *JHEP* **2014**, 07, 047, 1307.4787.
- [7] S. Abel, J. Rizos, *JHEP* **2014**, 08, 010, 1404.7359.
- [8] J. Halverson, B. Nelson, F. Ruehle, *JHEP* **2019**, 06, 003, 1903.11616.
- [9] M. Larfors, R. Schneider, *Fortsch. Phys.* **2020**, 68, 2000034, 2003.04817.
- [10] A. Constantin, T. R. Harvey, A. Lukas, 2108.07316.
- [11] A. Cole, A. Schachner, G. Shiu, *JHEP* **2019**, 11, 045, 1907.10072.
- [12] S. Krippendorf, R. Kroepsch, M. Syaeri, 2107.04039.
- [13] A. Constantin, A. Lukas, *Fortsch. Phys.* **2019**, 67, 1900084, 1808.09992.
- [14] D. Klaewer, L. Schlechter, *Phys. Lett.* **2019**, B789, 438, 1809.02547.
- [15] M. Larfors, R. Schneider, *Fortsch. Phys.* **2019**, 67, 1900083, 1906.00392.
- [16] C. R. Brodie, A. Constantin, R. Deen, A. Lukas, *Fortsch. Phys.* **2020**, 68, 1900087, 1906.08730.
- [17] C. R. Brodie, A. Constantin, R. Deen, A. Lukas, *Fortsch. Phys.* **2020**, 68, 1900086, 1906.08769.
- [18] C. R. Brodie, A. Constantin, R. Deen, A. Lukas, 1906.08363.
- [19] C. R. Brodie, A. Constantin, 2009.01275.
- [20] C. R. Brodie, A. Constantin, A. Lukas, 2010.06597.
- [21] J. Holland, *Adaptation in Natural and Artificial Systems*, The MIT Press reprinted, **1992**.
- [22] E. Goldberg, David, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, **1989**.
- [23] J. Holland, in *Toward a Practice of Autonomous Systems: proceedings of the first European conference on Artificial Life*, F. J. Varela and P. Bourgine, eds. MIT Press, **1992**.
- [24] T. Jones, *Evolutionary Algorithms, Fitness Landscapes and Search (PhD)*.
- [25] C. Reeves, J. Rowe, *Genetic Algorithms: Principles and Perspectives*, Springer, **2002**.
- [26] P. Charbonneau, *Tech. Rep. TN-450+IA*, National Center for Atmospheric Research, **Mar., 2002**.

- [27] R. Haupt, S. Haupt, *Practical genetic algorithms*, Wiley, 2nd ed., **2004**.
- [28] D. Michalewicz, *How to Solve It: Modern Heuristics*, Springer. Z Fogel 2nd ed., **2004**.
- [29] J. McCall, *J. Comput. Appl. Math.* **2005**, *184*, 205. Special Issue on Mathematics Applied to Immunology.
- [30] D. White, An Overview of Schema Theory.
- [31] P. Charbonneau, *ApJS* **Dec.**, **1995**, *101*, 309.
- [32] A. Yamaguchi, H. Nakajima, *Nucl. Phys. Proc. Suppl.* **2000**, *83*, 840, hep-lat/9909064.
- [33] T. S. Metcalfe, P. Charbonneau, *J. Comput. Phys.* **2003**, *185*, 176, astro-ph/0208315.
- [34] B. C. Allanach, D. Grellscheid, F. Quevedo, *JHEP* **2004**, *07*, 069, hep-ph/0406277.
- [35] Y. Akrami, P. Scott, J. Edsjo, J. Conrad, L. Bergstrom, *JHEP* **2010**, *04*, 057, 0910.3950.
- [36] S. Nesseris, J. Garcia-Bellido, *J. Cosmol. Astropart. Phys.* **Nov**, **2012**, *2012*, 0332033.
- [37] J. Blåbäck, U. Danielsson, G. Dibitetto, *JHEP* **2013**, *08*, 054, 1301.7073.
- [38] C. Damien, L. R. Diaz-Barron, O. Loaiza-Brito, M. Sabido, *J. High Energy Phys.* **Jun**, **2013**, 2013.
- [39] C. Damian, O. Loaiza-Brito, *Phys. Rev. D* **2013**, *88*, 046008, 1304.0792.
- [40] J. Blåbäck, U. Danielsson, G. Dibitetto, *JCAP* **2014**, *03*, 003, 1310.8300.
- [41] J. Blåbäck, D. Roest, I. Zavala, *Phys. Rev. D* **2014**, *90*, 024065, 1312.5328.
- [42] R. Hogan, M. Fairbairn, N. Seeburn, *Mon. Not. Roy. Astron. Soc.* **2015**, *449*, 2040, 1412.5997.
- [43] J. K. Behr, D. Bortoletto, J. A. Frost, N. P. Hartland, C. Issever, J. Rojo, *Eur. Phys. J. C* **2016**, *76*, 386, 1512.08928.
- [44] F. Ruehle, *JHEP* **2017**, *08*, 038, 1706.07024.
- [45] S. Abel, D. G. Cerdeño, S. Robles, 1805.03615.
- [46] K. Bull, Y.-H. He, V. Jejjala, C. Mishra, *Phys. Lett. B* **2018**, *785*, 65, 1806.03121.
- [47] A. E. Cole, *Identifying and Exploiting Structure in Cosmological and String Theoretic Data*, PhD thesis, U. Wisconsin, Madison (main), **2020**.
- [48] S. AbdusSalam, S. Abel, M. Cicoli, F. Quevedo, P. Shukla, *JHEP* **2020**, *08*, 047, 2005.11329.
- [49] N. Cabo Bizet, C. Damian, O. Loaiza-Brito, D. K. M. Peña, J. A. Montañez Barrera, *Eur. Phys. J. C* **2020**, *80*, 766, 2006.07290.
- [50] I. Bena, J. Blåbäck, M. Graña, S. Lust, 2103.03250.
- [51] A. E. Faraggi, in *Beyond Standard Model: From Theory to Experiment*, **2021**, 5. 2105.07970.
- [52] S. Forrest, M. Mitchell, *Mach. Learn.* **Nov**, **1993**, *13*, 285.
- [53] T. Jones, S. Forrest, in *Proceedings of the 6th International Conference on Genetic Algorithms*, L. J. Eshelman, ed. **1995**.
- [54] T. Jones, S. Forrest, Working Papers 95-02-022, Santa Fe Institute, **Feb.**, **1995**.
- [55] P. Collard, A. Gaspar, M. Clergue, C. Escazu, *Fitness Distance Correlation, as statistical measure of Genetic Algorithm difficulty, revisited*, **1998**.
- [56] P. Charbonneau, *Tech. Rep. TN-418+IA*, National Center for Atmospheric Research, **Dec.**, **1995**.
- [57] P. Charbonneau, *Tech. Rep. TN-451+STR*, National Center for Atmospheric Research, **Mar.**, **2002**.
- [58] A. Prügel-Bennett, J. L. Shapiro, *Phys. Rev. Lett.* **Feb**, **1994**, *72*, 1305.
- [59] J. Heredia, *Evol. Comput.* **11**, **2017**, 26, 1.
- [60] J. Distler, B. R. Greene, *Nucl. Phys. B* **1988**, *304*, 1.
- [61] S. Kachru, *Phys. Lett. B* **1995**, *349*, 76, hep-th/9501131.
- [62] L. B. Anderson, *Heterotic and M-theory Compactifications for String Phenomenology*, PhD thesis, Oxford U., **2008**. 0808.3621.
- [63] L. B. Anderson, Y.-H. He, A. Lukas, *JHEP* **2008**, *07*, 104, 0805.2875.
- [64] L. B. Anderson, J. Gray, Y.-H. He, A. Lukas, *JHEP* **2010**, *02*, 054, 0911.1569.
- [65] Y.-H. He, S.-J. Lee, A. Lukas, *JHEP* **2010**, *05*, 071, 0911.0865.
- [66] V. Braun, *JHEP* **2011**, *1104*, 005, 1003.3235.
- [67] A. Braun, A. Lukas, C. Sun, *Commun. Math. Phys.* **2018**, *360*, 935, 1704.07812.
- [68] P. Candelas, A. Constantin, C. Mishra, *Fortsch. Phys.* **2018**, *66*, 1800029, 1602.06303.
- [69] L. B. Anderson, J. Gray, A. Lukas, E. Palti, *Phys. Rev.* **2011**, *D84*, 106005, 1106.4804.
- [70] L. B. Anderson, J. Gray, A. Lukas, E. Palti, *JHEP* **2012**, *06*, 113, 1202.1757.
- [71] Mathematica, Version 12.2. Wolfram Research, Inc., Champaign, IL, **2020**.