Genotype-Phenotype Maps for Gene Networks: from Evolution to Computation

Francisco Quevedo Camargo
Brasenose College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Michaelmas 2017

I've been told that the average number of people
that read a PhD thesis is three.

So I dedicate this thesis to you, to you, and to you.

Acknowledgements

First of all, I would like to thank my supervisor, Ard Louis, for teaching me the joy and the importance of embracing the unknown, of being comfortable with the discomfort of being out of your depth, and of going after big questions. I have also been privileged to work alongside a number of brilliant colleagues and collaborators over these years: Stephanie Owen, Edward Rolls, Nora Martin, Benjamin Singer, Ambrose Yim, Matias Janvin and Guillermo Valle Pérez and Kamaludin Dingle, working with you has been a thrill and a pleasure.

I am also very thankful to Roberto Kraenkel, Frederico Gueiros-Filho, Ruth Baker, Philip Maini and Kevin Foster, for believing in me and giving me the opportunity to engage with scientific research, and for the Clarendon Fund, without which I probably would not have undertaken this D.Phil. I will also never forget the warm support of Elspeth Garman, who watched over me from day one.

I should also mention Ferdinando Randisi, the most outstanding housemate and officemate I could ask for, along with Megan Engel, who made our office at Physics a place where I always felt at ease. I am also grateful for the support of Scott Hale, Graham McNeill and Jonathan Bright in my final steps.

This would not be complete without mentioning some of the amazing people that I was lucky to meet in Oxford: Bayar Menzat, Inés Laura Dawson, Thomas Leissing, Jonathan Hadida, Jakub Tomek, Markéta Tomková, Billy Woods, Eric Haney, Franziska Kohlt, Samuel Forbes, Nils Reimer, Ananya Renuka Balakrishna, Stefano Ortona, Andreas Göbel, Lara Weisweiller-Wu, Rose Segal, Haden Spence, Erick Omena, Karen Haidinger, Juliano Spyer, Thais Rocha, Késia Decoté, Tamiris and Wesley Correa, Jessica Frazão, Dominique Santos, Eduardo Queiroz, Sergio Carvalho, Pablo Astudillo, Gustavo Quino, Thomas Bauwens, and many more. Equally important were my folks back home: Ana Ciconelle, Danilo Furlan Kaid, Viviane Santos, Otto Heringer, Catxerê Andrade Casacio, Cleandho Souza, Marina Magalhães, Pedro Pessoa, Silvana Kikuchi Oshiro, Macarena Lopez, Marcelo Camargo and Emilio Garcia have all made this an amazing ride.

I cannot find words to express my gratitude to Júlia Raíces, Rita Drummond, Citlali Solis Salas, Rita Nissim, Thais Sala, Ivan Cândido-Ferreira and Karine Yuki, who have kept me sane, and to my family, for their relentless love and support. Mom, Dad, Beatriz, you were my safe haven for when I lost my ground. Finally, the serendipitously wonderful Yayoi Teramoto, who has been an incredible source of support, advice, and magic.

Last but not least, I would like to thank Michelle Bosher, for making the Physics department a warm place to be, Russell Jones, for always being so helpful to someone who probably wanted to tweak his machine way too much, and Jonathan Patterson, for being *Deus ex machina* every time I accidentally deleted an important file.

Abstract

One of the most fundamental and least understood elements of evolution is the mapping between genotype and phenotype. Recent work on genotype-phenotype (GP) maps suggests that these maps show properties which may have important evolutionary implications. These properties include a skewed distribution of genotypes over phenotypes, linear scaling between phenotype robustness and the logarithm of phenotype frequency, and a positive correlation between phenotype robustness and evolvability. However, most of these properties have only been studied for self-assembling systems, such as protein complexes or RNA folding. In this thesis, we ask ourselves if these properties are more general.

First, we apply tools from algorithmic information theory to a wide class of input-output maps, of which GP maps are a subset. We find that these maps show a strong bias towards simple phenotypes, a pattern known as *simplicity bias*. We also define a *matrix map* of tunable complexity, with which we can study the conditions under which simplicity bias is present.

Next, we investigate multiple models of GP maps for gene regulatory networks (GRNs). These include Boolean threshold networks, where we fix the strength of gene interactions, while varying the network topology, as well as systems of differential equations, where we fix the network topology while varying interaction strengths. For both modelling frameworks, the GRN GP maps exhibit all the structural properties found in the literature, as well as simplicity bias. We also find that the number of genotypes mapping to the wild-type phenotypes for various GRNs is unusually large, and argue that this is evidence that the structure of the GP map plays an important role in determining evolutionary outcomes.

Finally, we return to more general input-output maps, and show that in addition to simplicity bias these maps also present *randomness deficiency*, that is, their output spectrum is less complex than expected. We argue that this additional property combines with simplicity bias in GP maps, and more generally, in input-output maps, and suggest a general trend towards simplicity in nature.

Contents

Chapter 1

Introduction

> "*Our ignorance of the laws of variation is profound. Not in one case out of a hundred can we pretend to assign any reason why this or that part differs, more or less, from the same part in the parents.*"
>
> — Charles Darwin, *On the Origin of Species*

1.1 Evolutionary biology into the twenty-first century

By the beginning of the twentieth century, forty years after Charles Darwin's famous book *On the Origin of Species* was published, Darwin's ideas were in the centre of a controversy. Darwinism, as it was called, was based on two principles. The first principle was that all life had come from a single ancestor; this idea had been proposed by others before, but Darwin was the first to collect a robust body of evidence to support it. The second principle was the source of the controversy: Darwin claimed that natural selection, the term he used to describe how differences in individual characteristics led to the differential survival and reproduction of individual organisms, was the main driver of evolutionary change. Darwin defined natural selection as the "*principle by which each slight variation [of a trait], if useful, is preserved*" [57], meaning that provided any form of inheritable variation in a population, the individuals with the most advantageous variations should survive and reproduce more. Darwin theorised that evolutionary change should be gradual, and any large change would be the result of cumulative small changes. The controversy arose because not all biologists were convinced that natural selection was the major agent of evolutionary change.

Admittedly, Darwin himself did not know where the variation needed for natural selection came from. In particular, he was not aware of Mendel's pioneer work on genetics, which was

rediscovered in 1900 by Hugo de Vries and Carl Correns [33]. According to Mendel's laws, genetic traits were inherited as discrete units, which raised questions about the possibility of gradual evolutionary change as proposed by Darwin. Mendelian inheritance, along with the newly discovered phenomenon of mutations, suggested evolutionary change was better explained by *saltationism*, the idea that evolution advanced in large leaps, as opposed to small steps.

The effort to solve this apparent incompatibility between Mendelian inheritance and Darwinism resulted in the birth of population genetics. This new field solved this impasse using tools from statistics to show that there was no contradiction between Mendel and Darwin. They demonstrated that the combined action changes to several discrete genetic units would produce distributions of quantitative characters (namely, the normal distribution), which could then be treated as continuous distributions. In this way, the small continuous changes that Darwin's natural selection required were derived from genetics itself. This combination of genetics and evolution, stemming from a series of papers by Sewall Wright [243], J. B. S. Haldane [103] and Ronald Fisher [81], made population genetics one of the key elements of the *modern synthesis* of evolutionary biology [110].

Although this merging of subjects into population genetics showed that Mendelian and Darwinian ideas were compatible, it did not end the discussion on the importance of selection when compared to other drivers of evolution. In 1968 Motoo Kimura proposed the *neutral theory of molecular evolution* [124, 125], which stated that much of evolution was driven by mutations that neither increased nor decreased the fitness of the organism. According to Kimura, changes in the frequency of a gene happened due to genetic drift, a statistical sampling effect that would always be present in finite populations. This neutral theory was strongly held against Ronald Fisher's work, which argued that genetic drift played a very minor role in evolution [81], and that selection was the main source of change and adaptation.

Despite Kimura and Fisher's disagreement on the role of random genetic drift, the modern synthesis was indeed a synthesis of many streams of evolutionary biology into one coherent framework. Nevertheless, the later decades of the twentieth century brought a series of conceptual developments that went beyond the modern synthesis, and that did not always agree with Darwin's original ideas [190]. One example was the discovery of horizontal gene transfer [180] and endosymbiosis [199], which are ways in which large evolutionary changes

can happen suddenly. Another well known late addition to the edifice of Darwinism is the field of evolutionary developmental biology (evo-devo), combining phylogenetics, gene regulation, and developmental biology [88]. The discovery of gene regulation had an important implication for evolutionary change, as it meant that a small modification in the genetic material of an organism could result in a large change in its observable traits. This reignited the debate of saltationism versus Darwinism. This debate is also present in Eldredge and Gould's theory of punctuated equilibria, which states that the gradual evolutionary change proposed by Darwin is not seen in the fossil record, and that evolution – in particular speciation – would happen through rapid jumps, followed by periods of little evolutionary change [94].

This growing list of developments has led to a push for an *extended evolutionary synthesis* [190]. There is much debate about whether this extension is necessary, as supporters of the modern synthesis hold that random genetic mutations and natural selection are enough to explain all the diversity observed in the natural world, while proponents of the extended evolutionary synthesis claim that these two mechanisms are not enough to explain evolution [130, 177]. This extended synthesis would include a series of concepts that were not mentioned by the original synthesis, such as multilevel selection [189], niche construction [131], high-dimensional fitness landscapes [190], epigenetics [204], evolvability [235], evo-devo [174] and nongenetic inheritance [31].

In a similar spirit to that of the extended evolutionary synthesis, in this thesis we aim to deepen our knowledge of what drives evolutionary change. At its most fundamental level, mutations change genotypes, which in turn determine phenotypes. This step from genotypes to phenotypes can be conceptualised as a genotype-phenotype map. These maps dictate how new variation arises, which natural selection can then subsequently act upon. In the modern synthesis, this map has typically been ignored [93]. We believe that these overlooked genotype-phenotype maps may in fact shed important light on the way that evolution works.

1.2 The history behind genotype-phenotype maps

While the question of how genotypes map to phenotypes has been a research topic since the rediscovery of Mendel at the beginning of the twentieth century, the term *genotype-phenotype*

map (GP map) first appeared in a paper by Jim Burns [36], who defined the GP map as the mapping from the genetic composition of an organism (its genotype) to its observable traits (its phenotype). Different faces of the concept of GP maps appeared the following decade, as John Maynard Smith proposed the concept of a 'protein space' comprising all possible amino acid sequences [211], and Richard Lewontin described the average genotype and phenotype of a population as points in the space of all possible genotypes and in the space of all possible phenotypes [144]. GP maps were popularised by Pere Alberch in 1991, as a conceptual tool to integrate genetics and developmental biology. Alberch proposed that the genotype-phenotype mapping function could be studied as a mathematical function from the space of genotypes to the space of phenotypes [8].

As mentioned above, the genotype represents the genetic composition of an organism, which is in its DNA. Genetic information can also be expressed in different forms by an RNA or amino acid sequence. In more abstract models, the genotype of an organism can be represented as a binary string [96] or as a set of Boolean functions [47], and most of the time it is represented as a discrete object. Phenotypes, instead, are defined in a broad, context-dependent way. For instance, for a genotype defined as an amino acid sequence, its phenotype could be defined at molecular level as the protein structure that results from the folding of the amino acid chain. At the level of metabolism, the phenotype might be defined as the function of the same protein. At the level of organism, the phenotype could be as simple as having grey or black fur. At larger scales, one has to include the interactions of that organism with its environment, and the definition of phenotype only becomes more complex. For this reason, much of the modern literature has focussed on simple mathematical models of GP maps where a relatively simple phenotype can be predicted with a certain degree of accuracy from its genotype – such as, for example, the GP maps shown in Figure 1.1.

Figure 1.1 presents three iconic examples of GP maps describing molecular self-assembly. At the top, we show the mapping from RNA sequences to their corresponding secondary structures. While the central dogma of molecular biology dictates that DNA is used to produce RNA which is used to produce proteins, not all RNA molecules encode proteins. A *non-coding* RNA molecule will typically form bonds between different parts of the molecule, and its secondary structure is defined as a configuration of bonded and non-bonded RNA bases corresponding to its folded structure in nature. While *a priori* a given RNA sequence might have more than one minimum folded structure, its structure typically minimises its

free energy for a given temperature – in this case, 37 °C. Molecules with different sequences may adopt different secondary structures, and perform different functions in the cell. RNA secondary structures can be computed with high accuracy using software such as the ViennaRNA package [208, 207].

In the middle of Figure 1.1 we show the HP model, a lattice model of protein folding, where a chain of amino acids adopts a conformation of least energy, much like for the RNA map. Since protein folding is a notoriously difficult problem to study, there are many two-dimensional and three-dimensional lattice models of protein structures. The HP model we present here is one of these simplified models, where sequences made of hydrophobic (H) and polar (P) amino acids fold onto a two-dimensional lattice [134].

At the bottom of Figure 1.1 we show the Polyomino model of self-assembly [7, 97]. In this model of protein quaternary structures, i.e. the structures formed by multiple proteins bound to each other, genotypes are represented by sequences which encode the interfaces of square tiles, which then self-assemble on a lattice.

Despite the many limitations of current GP maps, they have proven to be a useful lens in understanding the origins of phenotypic variation, and how this variation can steer evolution even before natural selection comes into play [233, 234, 202]. Much of this understanding comes from a growing body of research on *structural properties* shared by multiple GP maps [99, 233]. These properties are described in the next section, following a recent (2017) review by Sebastian Ahnert [6].

1.3 Structural properties of genotype-phenotype maps

Even though the specific mapping from a genotype to its corresponding phenotype is determined by the biology of its particular GP map, it turns out that there are many structural properties shared across maps. The name *structural* refers to properties that describe the statistical distribution of phenotypes across the point-mutation network of genotypes, as well as the topological properties of this network [6]. Three of these structural properties are illustrated in Figure 1.2.

1.3.1 Redundancy

For a GP map to have any of the other properties described in this section, it must first have more genotypes than phenotypes. This property, named *redundancy*, is a natural

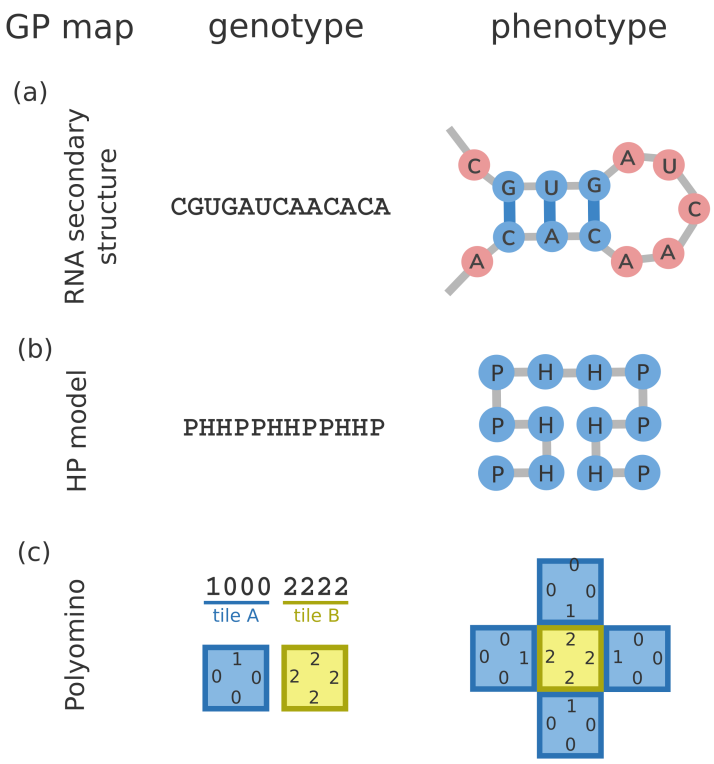


Figure 1.1: **An illustration of three GP maps, adapted from ref. [6]. (a)** the map from RNA sequence to secondary structure, which produces a folded secondary structure from a sequence of the four bases A, C, G and U [208, 207]. The RNA folding process can be predicted computationally with a high level of accuracy. **(b)** The HP model of protein folding [134]. The example shown here is a sequence of hydrophobic (H) and polar (P) amino acid residues that folds onto a two-dimensional lattice. **(c)** The Polyomino model [7, 97] is a self-assembly model in which sequences encode the interfaces of square tiles, which self-assemble on a lattice in a stochastic process. In this case, interface 1 binds to interface 2, and interface 0 does not interact. The two tiles self-assemble into the five-tile shape shown on the right.

extension of Kimura's description of the role of neutral mutations in evolution: if mutations are likely to have little or no fitness effect, it would not be unreasonable to imagine that some mutations to the genotype of an organism would leave its phenotype completely unchanged. Therefore, there should be more genotypes than phenotypes.

Although many GP maps show redundancy, the specific numbers of genotypes and phenotypes vary from map to map, in size and in scaling. For instance, for the mapping from RNA sequences of length L to their corresponding molecular secondary structures, the number of possible genotypes grows as 4^L, whereas the number of phenotypes grows as approximately 1.8^L, much more slowly than the number of genotypes [68]. While the exact ratio between the number of genotypes and the number of phenotypes depends on the

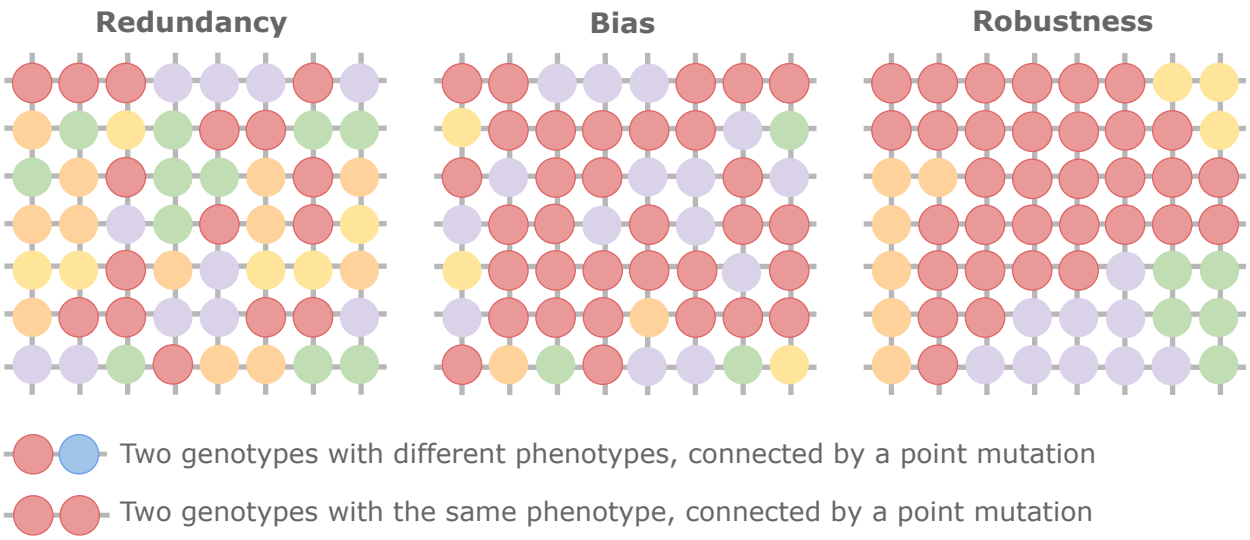


Figure 1.2: **Three structural properties of GP maps.** In the figures above, GP maps are illustrated as networks of of genotypes connected by single-point mutations and mapping to particular phenotypes, which are marked by different colours. On the left, the redundancy of a map indicates how the number of genotypes is much larger than the number of phenotypes. In the centre, the bias of a map indicates that some phenotypes correspond to many genotypes, while other phenotypes correspond to a few genotypes. In the GP maps discussed in this thesis, the numbers of genotypes mapping to each phenotype vary by orders of magnitude. On the right, robustness indicates that neighbour genotypes are more likely to correspond to the same phenotype than non-neighbour genotypes. As discussed in the text, redundancy is necessary, but not sufficient for bias and robustness. Adapted from Greenbury et al [99].

GP map (and on the system size, such as L for the RNA sequences), genotypes typically outnumber phenotypes by many orders of magnitude [6, 97].

The concept of redundancy is often expressed in terms of *neutral sets*, which are the regions of genotype space that map to the same phenotype. In discrete genotype spaces, when these regions are connected, they form *neutral networks* of genotypes connected by neutral mutations. A neutral network may range from not being connected at all to having a single connected component. The number of genotypes in a neutral network/set has been given different names in the literature, such as the phenotype's degeneracy level [32], abundance [54], designability [146, 178], genotype set size [192], neutral network size [208, 232], or neutral set size [202]. In this thesis, we will also use the word frequency, meaning the neutral network size of a phenotype divided by the size of the whole genotype space. The frequency of a phenotype can also be understood as the probability that a randomly chosen genotype will map to that phenotype.

1.3.2 Bias

In addition to being much more numerous than phenotypes, genotypes are also distributed over phenotypes in a very non-uniform fashion. Typically, a couple of phenotypes will take up most of genotype space, while most phenotypes will correspond to only a small fraction of the genotypes [6]. The bias of a GP map is the extent to which this distribution is uneven. For example, for the 6×6 HP model lattice protein map by Li et al. presented in Figure 1.1, 95% of genotype space is covered by 65% of phenotypes [146]. In comparison, for the map from RNA sequences to secondary structures discussed above, for $L = 20$ RNA, the same 95% of genotype space is covered by less than 10% of all phenotypes [97].

Phenotype neutral network sizes often span over orders of magnitude, resembling power laws or lognormal distributions. This pattern was recently captured in theoretical work by Susanna Manrubia and José Cuesta, who derived the distribution of neutral network sizes for a series of model GP maps, showing that strongly constrained sequences lead to power law distributions, while less constrained sequences lead to lognormal distributions [161]. Figure 1.3 shows the distribution of neutral network sizes for the lattice protein and RNA maps, which are both roughly lognormal.

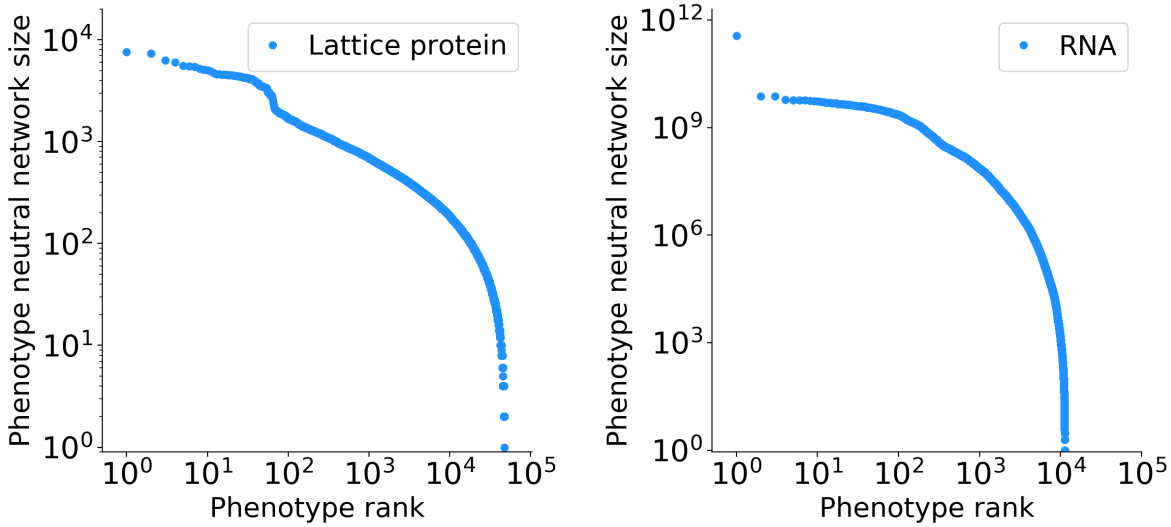


Figure 1.3: Distribution of phenotype neutral network sizes for two GP maps, lattice proteins in the 6×6 HP model (left) and $L = 20$ RNA (right). In both cases, neutral network sizes span over many orders of magnitude.

1.3.3 Robustness

Robustness is a measure of the ability of a system to resist perturbations. In biology, this idea comes under many names, such as buffering, canalisation, tolerance and developmental stability [233], which reflect the variety of perturbations an organism might suffer, from mutations altering its genetic sequence (mutational robustness) to variability in environmental conditions (environmental robustness). In the context of GP maps, the mutational robustness of a genotype is defined as the fraction of possible point mutations which leave its phenotype unchanged. One can also define the mutational robustness of a phenotype, also known as phenotype robustness, as the average of the genotype robustness over its neutral set [232].

A GP map where phenotypes are randomly distributed in genotype space is a completely *uncorrelated* map. In this map, for any given genotype, the probability that a neighbouring genotype maps to the same phenotype is simply given by the frequency of that phenotype in genotype space. Since a GP map typically produces many phenotypes of very low frequency (see subsection 1.3.2 above), the typical phenotype robustness can be quite low as well. For example, for the $L = 20$ RNA map, there are approximately 10^4 phenotypes, which implies that the mean phenotype frequency is on the order of 10^{-4}. If this map were completely uncorrelated, its average phenotype robustness would also be around 10^{-4}. This low robustness can be thought of as a null model expectation, where redundancy and bias are the only mechanisms in play.

However, in practice, measurements of phenotype robustness for different GP maps show that the null model does not hold: a phenotype's robustness is often much higher than its frequency. In many GP maps, phenotype robustness (ρ_p) scales linearly with the logarithm of phenotype frequency ($\log f_p$), as illustrated in Figure 1.4. The rough scaling of $\rho_p = a + b \log f_p$ seems to apply over many orders of magnitude and for different biological systems [96, 99, 6]. This robustness beyond the null expectation is a measurement of the correlations in a GP map [99]. A map where phenotypes have high robustness is a correlated map, as similar genotypes are likely to produce the same phenotype.

1.3.4 Evolvability

If robustness describes the ability of a phenotype to resist mutations, evolvability describes its ability to change and adapt. Its different formal definitions typically aim at quantifying

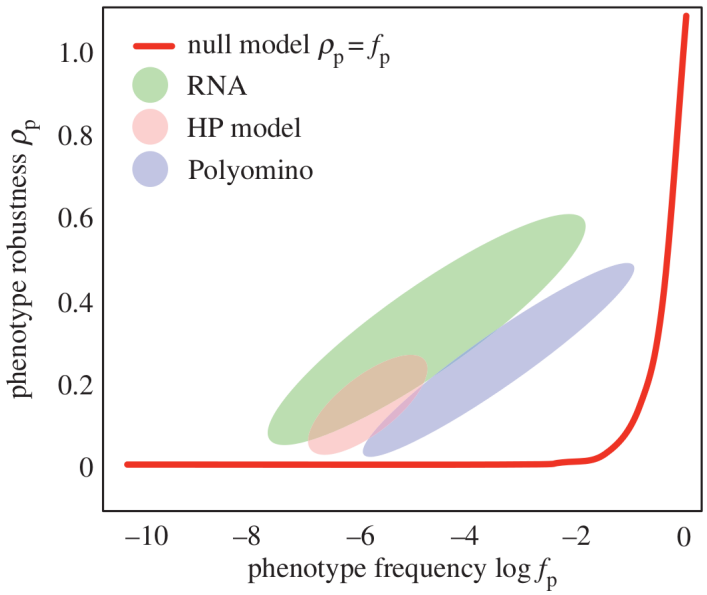


Figure 1.4: **Illustration of the relationship between phenotype robustness and frequency.** The plot shows the RNA, HP model and Polyomino GP maps, adapted from ref. [6]. For all maps, phenotype robustness ρ_p scales logarithmically with the phenotype frequency f_p. If the same phenotypes were distributed in genotype space following the null model described in the text, one would expect to see $\rho_p = f_p$, shown as a red line. The shaded ellipses show the location of most phenotypes for the three GP maps displayed. Full data in ref. [99].

the range of phenotypes that lie within a short mutation distance of a given genotype or phenotype [233, 56, 190]. Perhaps the most commonly used definitions are Andreas Wagner's definitions of genotype and phenotype evolvability [232]. Wagner defines the evolvability of a genotype as the number of different phenotypes in the point mutation neighbourhood of said genotype. Accordingly, the evolvability of a phenotype is defined as the number of different phenotypes which are one mutation away from its neutral network [232].

A priori, evolvability and robustness seem mutually incompatible, since mutations can be either neutral, which would result in higher robustness, or non-neutral, which would result in higher evolvability. That is exactly what happens at the genotype level: the lower the evolvability of a genotype, the higher its robustness. At phenotype level, however, the trend seems to reverse: in GP maps such as the RNA map [232] and the Polyomino map [97], there is evidence of a positive correlation between phenotype robustness and evolvability. This relation is illustrated in Figure 1.5. The conciliation between robustness and evolvability is crucial for evolution, as will be discussed later in section 1.3.6.

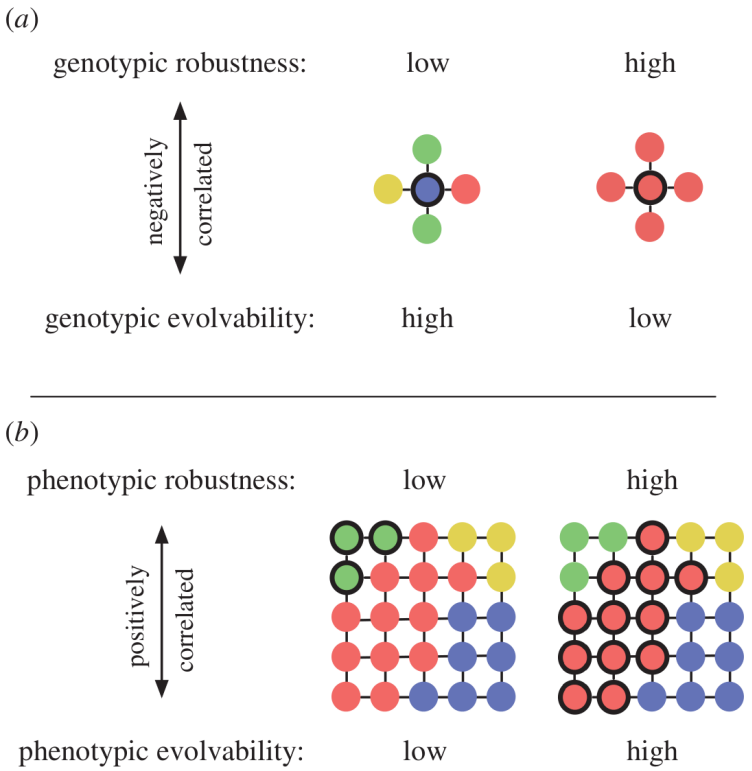


Figure 1.5: **Illustration of the relation between robustness and evolvability at genotype and at phenotype level, adapted from ref. [6]. (a)** At genotype level, both properties are mutually exclusive: if a genotype is one mutation away from many genotypes mapping to other phenotypes, its evolvability is high, and its robustness is low. If instead it is surrounded by many neutral neighbours, its robustness is high and its evolvability is low. **(b)** At phenotype level, there is no trade-off, and a phenotype can be being very robust, i.e. able to withstand many point mutations, and be very evolvable, i.e. be one point mutation away from many other phenotypes.

1.3.5 Shape space covering

Robustness and evolvability are not the only properties which affect the evolutionary search for different phenotypes: the combinatorial nature of GP maps also makes most phenotypes easily accessible from any point in genotype space. Looking again at the RNA map, even though the number of genotypes grows as 4^L, where L is the sequence length, no sequence is ever more than L point mutations away from any other. In fact, even L is not a tight upper bound, as most phenotypes can be reached within much less than L mutations [208]. This property, known as *shape space covering*, has also been observed for the HP lattice protein model [78] and for the Polyomino GP map [97].

1.3.6 Evolutionary consequences of GP map structure

These structural properties of GP maps, when combined, can strongly determine evolutionary outcomes. For instance, given a range of phenotypes of varying fitness, if the fittest phenotypes are confined to small neutral networks with only a few genotypes, they might never be discovered by evolutionary search. On the other hand, there are some indications that phenotypes with a large neutral network are likely to be found by evolution, regardless of their fitness [202], since a random mutation is likely to fall in a large neutral network. This effect has been named the "arrival of the frequent" [202].

The arrival of the frequent effect can be seen in its fullest form in the work of Dingle et al. [68], who study the RNA sequence-to-structure map discussed previously in this chapter. They find that the distribution of neutral network sizes of the RNA structures present in biological databases matches the distribution found by simply sampling genotype space, without any kind of selection. Considering that by sampling genotype space one can only find the phenotypes in the largest neutral networks, their study concludes that the phenotypes observed in nature are those with the largest neutral networks. As those phenotypes are a minute subset of all the possible phenotypes, this implies that biases in phenotypic variation strongly constrain the range of secondary structures available to natural selection. Fundamentally, this work shows an example of how strongly the structure of a GP map can affect evolutionary outcomes. This is a strong conclusion, which has only been seen for the RNA map so far.

Not only does the size of phenotype neutral networks affect evolution, but their internal structure can also have evolutionary consequences. The typical neutral network has a low average degree, a high clustering coefficient and a small diameter [5], all of which give the most frequent phenotypes a high robustness, while making it possible for a population to spread over genotype space without any cost in fitness. The topology of a neutral network also affects the time its corresponding phenotype would take to fix in a population, in a way that could significantly change the interpretation of molecular clock measurements [4, 202, 162, 160].

The correlated structure of GP maps also implies that one might not need natural selection to produce robust or evolvable phenotypes. Since it has been observed that robustness and evolvability are correlated at phenotype level [99, 241], and robustness is correlated

with phenotype frequency, the arrival of the frequent described above implies that robust and evolvable phenotypes will arise frequently, as a by-product of the uneven distribution of neutral network sizes. This is important as both robustness and evolvability are necessary for evolution, as a population needs to be able to withstand a degree of random mutations at genetic level, while remaining functional and evolving towards other phenotypes.

Interestingly, these structural properties of GP maps might help shed light on a question posed in 1966 by the mathematician Marcel Schützenberger, at a conference on challenges to the neo-Darwinian interpretation of evolution. He argued that random genetic mutations should be as inefficient in producing variation as randomly introducing mistakes in computer code – a process that would have little chance of producing any software that would run on a computer, let alone any software that could be a target of selection. According to Schützenberger, if biology behaved this way, evolution would never happen [205].

Yet, it does happen. In biological systems, random variation at genotype level and natural selection at phenotype level seem to be sufficient ingredients for evolution. The solution to this paradox, according to Schützenberger, lay in the step between sequence and function, between code and software:

> *[T]he question remains with respect to the relationships between the space of the chains of amino acids and the space of the organisms. (...) We do not know any general principle which would explain how to match blueprints viewed as typographic objects and the things they are supposed to control.*
> — Marcel Schützenberger [205], 1967, p.75

In his paper in 1966, Schützenberger pointed out that there was no known law or principle that allowed random mutations on genotypes to be any different from random mistakes in computer code. We believe the robustness observed in GP maps might provide such a principle.

The arrival of the frequent implies that random mutations often produce the same phenotypes, and the pervasive robustness of large neutral networks suggests an explanation for how evolutionary innovations can survive the variation introduced by random mutations. Besides, these structural properties of GP maps form a useful mathematical framework to study the evolution of biological processes that had were only discovered in the twentieth

century, while the modern synthesis was taking place. One example of such processes is gene regulation, discussed in the next section.

1.4 Gene regulatory networks

In this thesis, we use genotype-phenotype map lens to look at gene regulatory networks (GRNs), the networks of interacting molecules that control gene expression in a cell. GRNs have a central role in biological processes at all scales, such as regulating sugar processing in the single-celled baker's yeast, guiding cell proliferation in embryos, or orchestrating organ formation in macroscopic organisms. In essence, GRNs represent one of the ways in which information is transmitted inside inside living organisms, through the regulation of gene expression levels. This transmission of information can be understood as a gene network's phenotype. Its genotype, on the other hand, has to be a description of how the genes in the network interact – ultimately, a description of gene regulation.

1.4.1 Gene networks at the molecular level

Gene regulation is the process of passing information from a gene to another. It relies on a series of successful steps, starting with the synthesis of a functional gene product, a step also known as *gene expression*.

For gene expression to happen, the enzyme RNA polymerase must be able to bind to the DNA containing the gene to be expressed, producing an mRNA molecule that must then be able to bind to a ribosome, so that the mRNA can be translated into a sequence of amino acids. To ensure the first step of this chain of events happens, specific DNA sequences called *promoters* act as reliable binding sites where RNA polymerase can start transcription, which is the production of RNA from DNA [132].

Promoters also bind to *transcription factors*, sequence-specific DNA-binding proteins that can regulate the rate of transcription. Transcription factors work in a variety of ways, but the results of their actions can be summarised as either upregulating (also activating or enhancing) or downregulating (repressing, supressing) the transcription of a gene. Gene expression in prokaryotes is mostly regulated at the level of transcription [215], and eukaryotic gene regulation can also happen through acetylation and deacetylation of histone proteins, which results in exposing the DNA for transcription [89, 176], or even by post-translational modifications, such as phosphorylation or proteolysis [206].

Ultimately, the balance between positive and negative gene regulation comes from the differences in binding affinity between molecules. For instance, if there is any other molecule that binds more tightly to DNA than the RNA polymerase, or that binds more tightly to RNA than the ribosome, that molecule will act as a *repressor*, interrupting the sequence of transcription and translation. However, if the repressor is an *allosteric protein*, that is, if it changes conformation depending on which other molecules are bound to it, then the inhibition of gene expression can also be inhibited – which would then result in restoring gene expression. This is the case in the most classical and well-understood example of gene regulation, the *lac operon* in *Escherichia coli*, shown in Figure 1.6. In this genetic circuit, the expression of genes corresponding to lactose-digesting enzymes is normally turned off by an allosteric repressor, but lactose molecules can bind to the repressor and inactivate it by changing it to a non-DNA-binding conformation. Once the repressor is inactivated, RNA polymerase is able to bind to DNA, the lactose-utilisation genes are expressed, and the bacterium is able to digest lactose. The loop eventually closes, when all lactose is digested and the lactose-digesting genes go back to being suppressed.

Gene regulation is very important for the coordinated development of multiple cellular processes at the same time. If the same binding sequence is located in different regions of the genome, the same transcription factor will regulate multiple genes, making them respond to the same stimulus. For instance, if the binding sequence of the *lac* operon were present in many regions of the genome, all those regions would respond to the presence of lactose. The ability to regulate many genes at the same time is crucial for embryonic development, as it allows multiple cells to synchronously move, differentiate and alter their behaviour by changing which genes they express.

1.4.2 The topology of gene networks

When a number of genes influence the expression levels of each other, they form a network of transcription factors, or a gene regulatory network. Much work has been put into characterising the form and function of these networks, and today they are known to be sparse [135], modular [173], small-world [228] and scale-free [18]. All of these properties are believed to be a result of how gene networks grow: as a network is passed from an organism to its descendants, genes that are already connected to many other nodes in the network

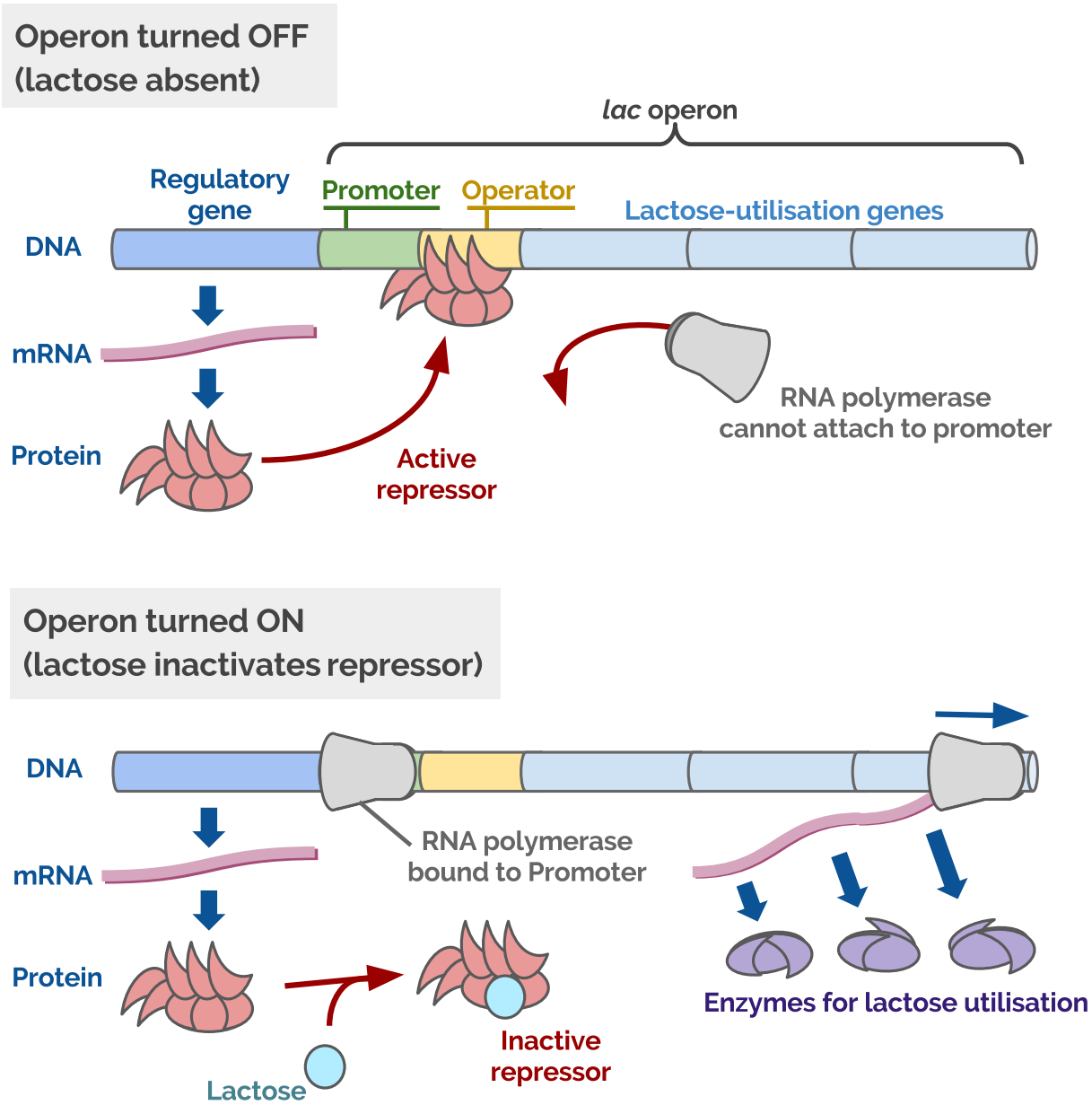


Figure 1.6: **Illustration of gene regulation in the *lac operon*. Top:** in the absence of lactose, the operon is repressed by a repressor which is the product of another gene. RNA polymerase then cannot bind to the promoter, and the lactose-utilisation genes are not transcribed. **Bottom:** when lactose is present, it inactivates the repressor, allowing RNA polymerase to bind to DNA and express the lactose-utilisation genes.

are likely to form more connections, resulting in a process of preferential attachment [18]. This process is likely to be mediated by gene duplication [23, 18].

Another remarkable aspect of gene networks is the occurrence of sub-networks known as *motifs*, such as feed-forward and feedback loops, in frequencies that differ from the frequencies of motifs in random networks [209, 173]. The reason for the higher frequencies of certain motifs in GRNs is still debated, as some researchers argue that the observed frequency of motifs in GRNs might simply be a product of an interplay between gene duplication and gene deletion [52, 157], and others argue that the frequency of certain motifs shows convergent evolution, and suggest it might be a product of natural selection [51].

This natural selection argument is also made regarding the overall gene network topology. Since highly connected networks come at a fitness cost [135], and scale-free networks are notoriously resistant to random failures of their nodes [50], the latter kind of network would be more robust to mutations. Scale-free networks also evolve easily towards oscillatory target outputs, which would also make them a desirable evolutionary outcome [98].

1.4.3 The evolution of gene networks

The evolution of gene networks is a strong determinant of the morphological differences between species. Non-coding sequences, i.e. DNA sequences that do not encode proteins, far from being "junk DNA", often contain gene regulatory sequences. This is particularly true for the *evo-devo gene toolkit*, a small set of genes which controls the embryonic development of an organism. Genes from this toolkit tend to go back in evolutionary history, being conserved over many phyla: for instance, all bilateral animals (vertebrates and insects, among others) share at least seven transcription factors involved in the regulation of embryonic development [83]. Besides, mutations or deletions of some of these highly conserved non-coding sequences are believed to be behind key differences between humans and other mammals [27, 100, 170].

1.4.4 Modelling gene networks

The non-trivial topology of gene regulatory networks together with the nonlinear relationships between their structure and the resulting gene expression make GRNs prime examples of complex systems in biology. As such, they have greatly benefited from mathematical modelling approaches, ranging from stochastic simulations close to molecular dynamics,

to time-continuous equations describing the kinetics of biochemical reactions, to Boolean functions where genes are reduced to switches that can be on and off.

At one end of the spectrum, gene expression can be modelled at the level of tracking every biomolecule in the system. This is typically done either using stochasatic simulations [90], or using coupled ordinary differential equations. In both cases, real variables $X_1(t), X_2(t), \ldots, X_N(t)$ express concentrations of the products of each gene, and interactions between those products are written as Michaelis-Menten-based equations of the form $X_i(t+\Delta t) = X_i(t) + f(X_1, \ldots, X_N)\,\Delta t$, where Δt represents a small time step and f is normally a combination of low-order polynomials and sigmoidal Hill functions [179, 224, 44].

At the other end, there is the Boolean approach pioneered by Stuart Kauffman [119, 115], in which genes are represented by nodes in a directed graph, with edges going from the genes that regulate to the ones that are regulated. In this graph, every gene can be either on or off, indicating whether it is being expressed or not. Time in this model advances in discrete steps, and the state of a gene at each time step is determined by a Boolean function on the state of the nodes regulating that gene. In other words, the expression level of a gene can only take two values, 0 and 1, and it depends on other genes through a predetermined Boolean function, such as `AND`, `OR`, or a combination of similar logical functions.

The range of modelling approaches also covers many models of intermediate complexity, such as differential equations with added delay [198], or discrete-time networks with continuous levels of gene expression, resembling models for artificial neural networks [231]. In this thesis, however, we will focus on differential equations and Boolean networks.

1.4.5 GP maps for gene regulatory networks

Mathematical models of GRNs have been largely employed to study the GP map from a GRN's structure to its behaviour. In work by Andreas Wagner's group, the structure of a gene network has been represented as a weighted adjacency matrix [47, 48], and alternatively as a set of Boolean functions determining the expression state of a gene as a function of other genes in the network [183, 184, 182], while its phenotype has been represented by the gene expression steady-state of the network [47, 182]. In the same way as the previous GP maps, these maps show a biased distribution of genotypes over phenotypes, and a trade-off between genotype robustness and evolvability [233]. Ibañez-Marcelo et al. [111] also provide a model of a GP map for GRNs, incorporating both the network topology and its initial state

of gene expression in its genotype definition, and also defining phenotypes in terms of the gene expression steady-state. This alternative definition of GP map also shows robustness and evolvability, as well as shape space covering [111]. In the next chapters, we will be discussing other ways to define GP maps for gene networks.

1.5 Thesis outline

This thesis will progress as follows. In Chapter 2, we present a quick review of results from algorithmic information theory, as well as an extension of some of those results to finite input-output maps. We then discuss their application to genotype-phenotype maps, and present the concept of simplicity bias, which describes when an input-output map is biased towards producing outputs that have low algorithmic complexity. We also introduce the matrix map, an input-output map with binary strings as inputs and outputs, where it is possible to tune the amount of information contained in the map, that is, the complexity of the map. In studying the matrix map, we find that not only do low-complexity or simple matrix maps produce a more biased distribution of neutral network sizes, but that simple maps are also more likely to show simplicity bias than complex maps.

In Chapter 3, we examine the properties of dynamical systems on Boolean threshold networks, and look into their application in studying GP maps for gene regulatory networks. We define three different models for the mapping between gene network connectivity and behaviour. The Boolean threshold network GP map exhibits a series of properties that are observed in other input-output maps in the literature, including simplicity bias. We also modify Jörg et al's neutral network size estimator algorithm [112] for use beyond its original scope, turning it into a powerful tool in the study of vast genotype spaces for GRNs. We then use this algorithm to investigate the neutral network sizes of wild-type gene networks.

In Chapter 4 we study gene regulatory networks as systems of ordinary differential equations. In contrast with the Boolean picture painted in Chapter 3, where time proceeds in discrete increments and gene interactions are restricted to binary values, in Chapter 4 we allow time to vary continuously, fix the gene network topology and vary the strength of gene-gene interactions. We then construct two different models for the GP map between network structure and behaviour, following different definitions of phenotype. We verify that these maps present the same structural properties as the GP map models discussed

in Chapter 3, which suggests these findings do not depend on the mathematical formalism used to describe this GP map.

For both the Boolean threshold network maps in Chapter 3 and the ordinary differential equation maps discussed in Chapter 4, we find that wild-type GRN phenotypes always have unusually large neutral network sizes. This suggests that the range of GRNs observed in nature is shaped by the arrival of the frequent effect, implying that the structure of this GP map is critical for understanding the evolution of GRNs. This finding also places gene networks as the first GP map besides the RNA sequence-to-structure map for which this effect has been observed.

In Chapter 5, we discuss the presence of randomness deficiency in input-output maps, defining it as the average difference between the complexity of the outputs of a given map and the complexity of a random string of same length. We show how this effect can act in combination with simplicity bias to produce phenotypes of low complexity in nature.

All maps and formalisms are discussed in the conclusion, where gene networks and matrix maps are brought together with other input-output maps, shining a light on the parallels between evolution and computation.

Chapter 2

Algorithmic Information Theory

> "*Algorithmic information theory is the result of putting Shannon's information theory and Turing's computability theory into a cocktail shaker and shaking vigorously.*"
>
> — Gregory Chaitin, *Information and Randomness*

2.1 Information theory in biology

The formal study of genotype-phenotype maps, abstracted away from specific biological contexts, allows them to be examined under a different perspective: information theory. A GP map can be seen as a machine that maps a set of inputs (genotypes) to a set of outputs (phenotypes), transforming questions about phenotype neutral networks, robustness or evolvability into questions about the inputs and outputs processed by this machine.

The information-centric approach is not new to biology. Shannon's information theory has a history of applications to bioinformatics and molecular biology [1, 163, 203, 2], and has also been used to study the thermodynamic efficiency of biological processes such as biochemical reactions [181], cell division [73] and gene expression [24]. Naturally, information theory has also been the framework used to study how biological systems might have evolved to gather, represent and process information [219], in systems ranging from DNA sequences [20] to populations of neurons [219] and even to populations of organisms, such as flocks of birds [25]. GP maps, as the intermediates between the information contained in genotype and phenotype, can also be studied a form of information processing — essentially, as input-output maps.

2.2 Basics of algorithmic information theory

Despite its wide application to the biological sciences, Shannon information theory plays a small role in this thesis. Instead, we employ tools from the field of algorithmic information theory (AIT), which was founded by Ray Solomonoff [213], and further developed by Andrey Kolmogorov [126], Gregory Chaitin [39], Leonid Levin [142, 143] and Per Martin-Löf [164]. In contrast with Shannon's information theory, AIT does not study the information content of random processes or probability distributions. Instead, AIT looks at discrete objects, such as finite strings, and at relations between sets of inputs and sets of outputs, which we here call input-output maps.

AIT is placed at the crosstalk between information processing and computability theory, a branch of mathematics and of the theory of computation that was born from the work of Kurt Gödel, Alonzo Church and Alan Turing. Computability theory is the study of *computable functions*, that is, functions for which there is an algorithm that can translate a set of inputs to their corresponding outputs [222]. The *Turing-Church thesis*, also known as the Turing-Church conjecture [222, 45, 46], states that a function on the natural numbers is computable by an algorithm if and only if it is computable by a Turing machine, a generic computation device proposed by Turing in his famous 1936 paper *On computable numbers, with an application to the Entscheidungsproblem* [222]. The Turing-Church thesis is important here, as it places Turing machines as the set containing all computable input-output maps: any computable relation between input and output should be equivalent to a Turing machine.

2.2.1 Kolmogorov complexity

The information content or *complexity* of a discrete object is a concept central to AIT. In Chaitin's words, *"the basic idea is to measure the complexity of an object by the size in bits of the smallest program for computing it"* [37]. Consider, for example, the following binary strings:

$$x_1 = 01010101010101010101010101010101$$

$$x_2 = 10011010110100110100101101000 1$$

To the human eye, the first string is clearly simple, and could be described as "`print 01 fifteen times`". The second string, on the other hand, shows no obvious pattern, and cannot be described with anything shorter than "`print 10011010110100110100101101000 1`".

In AIT, the information content or *Kolmogorov complexity* or algorithmic complexity of a string x is given by the length of its shortest description, or the length of the shortest program (or programs) that can generate x. Formally, it is defined as

$$C_U(x) = \min_p \{l(p) \ : \ U(p) = x\} \tag{2.1}$$

where $l(p)$ is the length of a binary program p in bits, and U is a *universal Turing machine* (UTM), a computing device that can simulate any Turing machine on any arbitrary input [222]. $C_U(x)$ is the length of the shortest program (or programs) from all programs that output x and then halt.

In practice, the subscript U is often dropped because of the *invariance theorem* [148, 53]. This theorem states that if one defines $C_U(x)$ and $C_V(x)$ as Kolmogorov complexities with respect to UTMs U and V respectively, then $|C_U(x) - C_V(x)| \leq c$ for any string x, where c is a constant that depends on U and V, but not on x. This can also be written as $C_U(x) = C_V(x) + \mathcal{O}(1)$, where $\mathcal{O}(1)$ denotes terms that are asymptotically independent of x, and which can be neglected in the limit of large x. In that limit, one speaks simply of the Kolmogorov complexity $C(x)$.

Kolmogorov complexity can also be used to define the AIT equivalents of conditional, joint and mutual information. In particular, the conditional complexity $C(x|y)$ can be interpreted as the length of the shortest program (or programs) that, when fed to a UTM that is also provided y, generates x and halts. $C(x|y)$ obeys the following relations:

$$C(x|y) \ \leq \ C(x) + \mathcal{O}(1) \tag{2.2}$$

$$C(x) \ \leq \ C(x|y) + C(y) + \mathcal{O}(1) \tag{2.3}$$

The first inequality comes from the fact that by providing the UTM with extra information one might make it easier to generate x. The conditional Kolmogorov complexity will range from $C(x|y) = O(1) << C(x)$, when y contains most of the information necessary to generate x (e.g. if x is just the y repeated a few times), to $C(x|y) = C(x) + \mathcal{O}(1)$ if y does not contain any information about x.

The second inequality stems from the definition of $C(x)$ as the length of the shortest program that generates x: if the program that generates y and then x results to be shorter than a program that only generates x, by definition the length of the former program would be the Kolmogorov complexity of x.

2.2.2 Most strings are complex

Most strings have a Kolmogorov complexity close to their length in bits. In other words, most strings are *uncompressible*: there is no way to describe them that is shorter than the strings themselves. This result comes from a counting argument known as the *pigeonhole principle*. Consider, for instance, the set $\{0,1\}^n$, denoting all 2^n binary strings of length n, and the set $\{0,1\}^{<n}$, denoting all $2^n - 1$ binary strings shorter than n. Since the latter set is one string smaller than the former set, there is no way to map all strings in $\{0,1\}^n$ to strings in $\{0,1\}^{<n}$, meaning that at least one of the strings of the former set will inevitably be uncompressible. In fact, since 2^{n-1} strings of the latter set are $(n-1)$-bits long, it means that at least half of the strings in $\{0,1\}^n$ cannot be compressed by more than one bit. More generally, the fraction of $\{0,1\}^n$ that can be compressed by k bits is bounded by 2^{-k}, and the fraction of strings that can be compressed by k bits or more is then $2^{-k+1} - 2^{-n}$.

The fact that most strings are uncompressible is easier to notice as n grows: for strings of $n = 100$, over 99.9% of all strings cannot be compressed by more than $k = 10$ bits, i.e. 10% of their length. For $n = 1000$, the same 99.9% can only be compressed by up to 1% of their length – the same 10 bits. Strings as such, that cannot be compressed by more than a few bits, are called complex, or *algorithmically random*. Since most strings behave in such way, a random string is likely to be algorithmically random. In terms of Kolmogorov complexity, this means that most strings x will have $C(x) \approx n$.

2.2.3 Prefix-free complexity and the coding theorem

In addition to $C(x)$, one can also define the *prefix-free complexity* $K_W(x)$ of a string x as:

$$K_W(x) = \min_p \{l(p) \ : \ W(p) = x\} \tag{2.4}$$

where $l(p)$ is the length of a binary program p and W is a prefix UTM, i.e. a universal Turing machine where an input program p is not the prefix (i.e. the first $l(p)$ bits) of any other input program [142, 39]. In practice, this means that *prefix-free* programs do not need

any spacer, end-of-line, or indication of the beginning or end of a program. The invariance theorem and the conditional complexity inequalities also hold for the prefix-free complexity, allowing us to drop the W and simply write $K(x)$ and $K(x|y)$ [148].

The plain Kolmogorov complexity $C(x)$ and the prefix-free complexity $K(x)$ are very close in value, diverging only in $\mathcal{O}(\log(C(x))$ [148]. Prefix codes can be used to define the *algorithmic probability* or universal probability $P(x)$ of an output:

$$P(x) = \sum_{p\,:\,U(p)=x} 2^{-l(p)} \tag{2.5}$$

In equation (2.5) above, the sum is over all binary input programs p which when run on a universal prefix Turing machine U will halt and output x. On reason to use a prefix UTM is Kraft's inequality [127], which states that if $\mathcal{P}$ is a set of binary prefix-free code words, then $\sum_{p\in\mathcal{P}} 2^{-l(p)} \leq 1$, which implies $\sum_x P(x) \leq 1$. Moreover, as $K(x)$ is defined as the length of the shortest program (or programs) that can output x when fed into a prefix UTM, the largest term of the sum is by definition $2^{-K(x)}$. This also places a lower bound on the probability of an output x, as it implies that $P(x) \geq 2^{-K(x)}$.

In 1974, L. A. Levin stated a stronger result, known as the *coding theorem* [142]:

$$2^{-K(x)} \leq P(x) \leq 2^{-K(x)+\mathcal{O}(1)} \tag{2.6}$$

Equation (2.6) holds for asymptotically long strings, and can also be read as $P(x) = 2^{-K(x)+\mathcal{O}(1)}$. This theorem implies that simple – low $K(x)$ – outputs have a higher probability, and complex – high $K(x)$ – outputs are less likely, with their probability falling exponentially with $K(x)$. In this way, the notion of algorithmic probability reflects the intuitive idea that patterns which result from simple processes should be likely, whereas patterns that can only result from very complex processes should be unlikely.

2.2.4 Kolmogorov complexity is uncomputable

In spite of their mathematical elegance and generality, results from AIT such as the coding theorem have seen little application to real computing systems. This is partly due to how these theorems rely on particular properties of universal Turing machines and have only been proven in the asymptotic limit of long binary strings, while real-world input-output machines or input-output maps typically deal with finite strings, and often do not have the computational power of a Turing machine.

Another reason behind the a priori inapplicability of AIT lies in the definition of Kolmogorov complexity: it is the length of the shortest input program that will output x *and then halt*. This means that an algorithm for $K(x)$ will involve determining whether an arbitrary program will run and halt, or whether it will never stop, a problem that is known as the *halting problem*. In his 1936 paper, Turing proved that there is no algorithm to solve the halting problem in Turing machines [222], which implies that Kolmogorov complexity is also uncomputable: there is no algorithm that will take x as an input and calculate $K(x)$ [148].

There are physical systems which have been mapped to UTMs, and physical measurements which are uncomputable like Kolmogorov complexity or undecidable like the halting problem [153, 55], but these will not be covered in this work. Most input-output machines used in science in engineering are in fact computable, having no halting problem: they will produce an output and then halt, for all valid inputs. One example of conputable input-output map is the RNA sequence-to-structure map mentioned in Chapter 1: to produce this map, we use the Vienna package for computational folding [106], which outputs a molecular secondary structure for every RNA sequence that it receives as input, and never keeps running indefinitely. All the GP maps we present in the next chapters are also computable.

2.3 The coding theorem for computable maps

Normally, the scope of AIT theorems does not extend to computable input-output maps, which are the maps that map all inputs to all outputs without halting. That said, it is possible to extend Levin's coding theorem to take them in consideration. Kamaludin Dingle [66] has derived an upper bound fo $P(x)$ for any computable function, namely:

$$P(x) \leq 2^{-K(x|f,n)+\mathcal{O}(1)} \tag{2.7}$$

In the equation above, $K(x|f,n)$ represents the complexity of an output x, given a computable map f and an integer n, which defines the length of the input strings, and therefore the size of the whole input space I. For binary strings, for example, $|I| = 2^n$ possible input strings. This equation has appeared in different forms in refs. [148] and [85], and we also derive it in ref. [67]. The derivation is reproduced here, as follows.

Consider a computable function $f : I \to O$, mapping the input space I, of size defined by n, to the output space O. Consider then the algorithm $\mathcal{A}$, defined by:

1. Enumerate all inputs of f, using n

2. Map all inputs to their corresponding outputs, using f
3. Print the resulting list of outputs x, as well as their corresponding probabilities $P(x)$ (i.e. their frequency in I)

Since $\mathcal{A}$ is provided with f and n, its complexity is $K(\mathcal{A} \mid f, n) = \mathcal{O}(1)$. It might seem counter-intuitive that applying an algorithm to the whole set I could have a complexity as low as $\mathcal{O}(1)$, but this result comes from the well known fact from AIT [53] that the procedure of enumerating all objects in a set can be much simpler than generating a typical element of the set. For example, take the set $\{0,1\}^n$ of all binary strings of length n. A typical element x of the set will be algorithmically random, meaning $K(x) \approx n$. On the other hand, to construct the set, one only needs up to $\log_2(n)$ bits to specify n, plus $\mathcal{O}(1)$ to print the whole set. With that in mind, it is not surprising that $K(\mathcal{A} \mid f, n) = \mathcal{O}(1)$.

Since an output x might be produced multiple times for different inputs, one can produce a compressed representation of the list of outputs and their corresponding probabilities $P(x)$. The process of turning each output into a codeword is called coding or encoding. For instance, outputs can be efficiently encoded using a Shannon-Fano-Elias (SFE) coding [53], which produces prefix-free binary words $E(x)$ for every output x. The length $l(E(x))$ of the binary codeword $E(x)$ varies with the probability $P(x)$ of the output, according to the equation (2.8) below, where $\lceil \cdot \rceil$ indicates the ceiling function:

$$l(E(x)) \;=\; \left\lceil \log_2 \left(\frac{1}{P(x)} \right) \right\rceil + 1 \tag{2.8}$$

So far, the steps we have described take as input the function f and an integer n, and produce outputs that are encoded using $l(E(x)) + \mathcal{O}(1)$ bits, where $\mathcal{O}(1)$ takes into account the fixed size of the program used to generate the SFE coding. Now, since the definition of Kolmogorov complexity is the length of the shortest description possible for a given UTM (up to $\mathcal{O}(1)$ terms), this implies that $K(x|f,n)$, which is the number of bits necessary to describe a given output x, given the map f and n, cannot be larger than the description derived using the SFE coding. In other words,

$$K(x|f,n) \leq l(E(x)) + \mathcal{O}(1) \tag{2.9}$$

$$= \log_2 \left(\frac{1}{P(x)} \right) + \mathcal{O}(1) \tag{2.10}$$

$$\Rightarrow P(x) \;\leq\; 2^{-K(x|f,n)+\mathcal{O}(1)} \tag{2.11}$$

Above, for ease of notation, we have used f to represent both the *function* that maps inputs to outputs and the *program* that implements it. The same applies for n, which stands for the number that defines the size of input space, as well as for the program to calculate n. In practice, since most n are uncompressible, the size of such a program will be $\log_2 n + \mathcal{O}(1)$.

Differently from Levin's coding theorem for UTMs, this version of the coding theorem for computable maps does not include a lower bound on $P(x)$, only an upper bound. Despite this limitation, it is possible to show [66, 67] that most inputs map to outputs which lie close to the upper bound. More specifically, we can derive a probabilistic lower bound, for $r \in (0, \infty)$:

$$\frac{2^{-K(x|f,n)+\mathcal{O}(1)}}{\mathcal{E}r} \leq P(x) \tag{2.12}$$

The reason this bound is probabilistic is that, while the upper bound given by equation (2.11) always holds, the lower bound in equation (2.12) is only true for a fraction p of all outputs, where $p \geq 1 - \frac{1}{r}$. For $r \to \infty$, the lower bound approaches the trivial bound $P(x) \geq 0$, which is satisfied by all outputs, and reflected in $p \to 1$. On the other hand, low values of r provide a higher lower bound, which will be satisfied by a smaller fraction p of the outputs.

The number $\mathcal{E}$ in equation (2.12) is defined as

$$\mathcal{E} \ = \ \sum_{i=1}^{N_O} 2^{-K(x_i|f,n)} \tag{2.13}$$

and the sum is over all outputs, indicated by N_O. The number $\mathcal{E}$ can also be thought of as a weighted sum of the ratio $\frac{2^{-K(x_i|f,n)}}{P(x_i)}$, weighing each output by its probability $P(x_i)$. In this way, $\mathcal{E}$ represents how far the outputs x are from the upper bound, on average. In ref. [67], we measure $\mathcal{E}$ for a series of input-output maps, and find that typically $\log_{10} \mathcal{E} \approx 1$ or 2. Given that the output frequencies $P(x)$ tend to vary over multiple orders of magnitude, we consider equation (2.11) to be a relatively tight upper bound for outputs produced from random inputs.

2.4 Upper bound for limited complexity maps

Equation (2.11) above applies to any computable map, but this generality can be a problem: since the upper bound on $P(x)$ depends on $K(x|f, n)$, different mapping functions f might

result in very different values for $K(x|f,n)$, leading to different bounds for $P(x)$. The solution we employ here is to restrict the prediction in equation (2.11) to input-output maps of *limited complexity*, defined as maps where $K(f) + K(n) \ll K(x) + \mathcal{O}(1)$ holds asymptotically [67]. This assumption, when combined with the properties of conditional Kolmogorov complexity presented in equations (2.2) and (2.3) holds the following expression:

$$\left.\begin{aligned} K(x) &\leq K(x|f,n) + K(f) + K(n) + \mathcal{O}(1) \\ K(x|f,n) &\leq K(x) + \mathcal{O}(1) \\ K(f) + K(n) &\ll K(x) + \mathcal{O}(1) \end{aligned}\right\} \Rightarrow K(x) \approx K(x|f,n) + \mathcal{O}(1) \quad (2.14)$$

The result above also shows that input-output maps of limited complexity do not present the situation $K(x|f,n) \ll K(x)$ discussed above. For maps of limited complexity, any variation on the upper bound on $P(x)$ can only be ascribed to differences between the outputs themselves.

Moreover, while the set of possible outputs is still defined by f and n, since $K(x) \approx K(x|f,n) + \mathcal{O}(1)$, for these maps the expression for the probability upper bound (2.11) becomes independent of f and n:

$$P(x) \lesssim 2^{-K(x)+\mathcal{O}(1)} \quad (2.15)$$

Our arguments above invoke $K(f) \ll K(x)$ and $K(n) \ll K(x)$ for a typical x. Most examples presented later in this section are maps of fixed complexity, i.e. $K(f) = \mathcal{O}(1)$, for which one can always find large enough x so that these inequalities hold. Since $K(n)$ scales asymptotically as $\mathcal{O}(\log n)$, it would seem that equation (2.15) should also hold for maps that scale as $\mathcal{O}(\log n)$, expanding the scope of equation (2.15) beyond fixed maps. On the other hand, if $K(f)$ scales as $\mathcal{O}(n)$, which is how $K(x)$ scales, or even faster than $K(x)$, say $\mathcal{O}(n^2)$, the inequality $K(f) + K(n) \ll K(x) + \mathcal{O}(1)$ no longer holds. In section 2.7, we present an example of an input-output map which scales as $\mathcal{O}(n^2)$, and therefore is not of limited complexity.

2.5 Approximating Kolmogorov complexity

The most influential complexity measure for digital strings (or sequences) was introduced in 1976 by Lempel and Ziv [138]. The Lempel-Ziv (LZ) family of algorithms is the foundation of many widespread compression schemes used today, including compressed file formats such as

GIF, PNG and ZIP [65]. The principle behind the Lempel-Ziv algorithm is to read through a string (of any finite alphabet size) from left to right and create a list of sub-patterns as they appear in the string. A string with many different sub-patterns would then be converted into a large list, and hence be assigned a high complexity. Conversely, a string that is essentially built up of a few repeated sub-patterns would show little variation, resulting in a short list of sub-patterns, and in a low complexity. Lempel and Ziv show that, for an ergodic source and in the limit of long sequences, the number of distinct words (sub-patterns) in the list given by $N_w(x)$ obeys the following equation for nearly all sequences [252, 139]:

$$\lim_{n\to\infty} \frac{N_w(x)\log_2(n)}{n} = \frac{K(x)}{n} = S(x) \tag{2.16}$$

where $K(x)$ stands for the Kolmogorov complexity of the string x, n is the length of the binary strings, and $S(x)$ is the standard Shannon entropy *rate*. The Lempel-Ziv complexity is a popular choice for approximating Kolmogorov complexity, and it is believed to work better than other lossless compression-based measures for shorter strings [13, 140].

We use the following approximate complexity measure based on the 1976 LZ algorithm [138]:

$$C_{LZ}(x) = \begin{cases} \log_2(n), & x = 0^n \text{ or } 1^n \\ \log_2(n)[N_w(x_1...x_n) + N_w(x_n...x_1)]/2, & \text{otherwise} \end{cases} \tag{2.17}$$

The special treatment given to 0^n and 1^n is due to the fact that $N_w(x)$ assigns complexity $K = 1$ to the strings `0` and `1`, but complexity $K = 2$ to any larger string made only of zeros or ones, whereas the Kolmogorov complexity of such a trivial string actually scales as $\log_2(n)$, as one needs to encode n. In this way we ensure that our $C_{LZ}(x)$ measure not only gives the correct behaviour for complex strings in the $\lim_{n\to\infty}$, as shown in equation (2.16), but also the correct behaviour for the simplest strings. In addition to the $\log_2(n)$ correction, taking the mean of the complexity of the forward and reversed strings makes the measure more fine-grained, since it allows more values for the complexity of a string. Note that $C_{LZ}(x)$ works just as well for strings made from alphabets with more than two symbols.

It is instructive to compare our complexity measure $\tilde{K}(x) = C_{LZ}(x)$ to the simple binary entropy, defined as $S(x) = p\log p + (1-p)\log(1-p)$, where p is the fraction of 1s (or 0s) in the string x. While low entropy strings typically have low complexity, the converse is not always true. Strings that are algorithmically simple can still have high entropy, as we

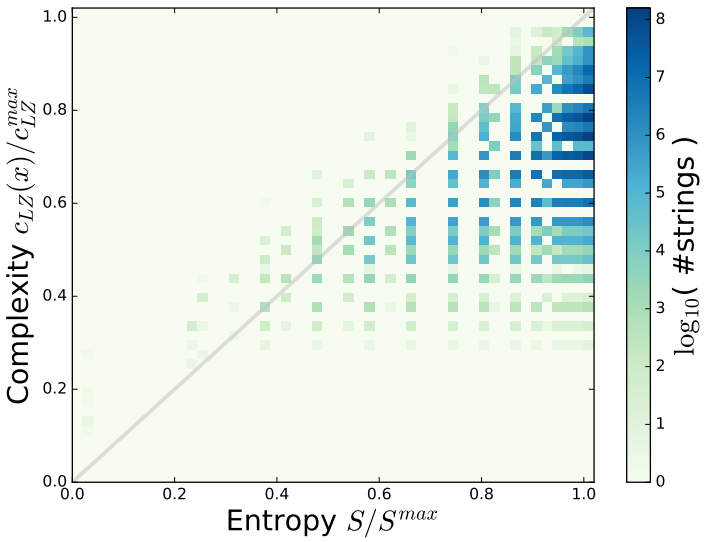


Figure 2.1: **Heatmap for the complexity $C_{LZ}(x)$ versus entropy $S(x)$ for binary strings of length $n = 30$.** Both measures are normalised by their maximum value. Complexity is bounded by entropy, in the sense that a binary string with mostly zeros (and therefore low entropy) cannot be complex. Conversely, a string such as $x = 010101\ldots01$ has maximum entropy, since it is made of an equal number of 0s and 1s (thus $S/S^{max} = 1$), while still being very simple ($C_{LZ}(x)/C_{LZ}^{max} \approx 0.273$).

illustrate in Figure 2.1. Nevertheless, as is well known in the literature, in the limit of long strings, the mean Kolmogorov complexity per length tends to the entropy rate – see also equation (2.16).

In Appendix A, we compare the Lempel-Ziv complexity with other complexity estimates, as well as entropy, and show that the phenomenon of simplicity bias, discussed in the next section, does not depend on the particular details of any complexity estimator.

2.6 Simplicity bias in input-output maps

In work in collaboration with Kamaludin Dingle and Ard A. Louis [67], we have applied this extension of the coding theorem to examples of discrete input-output maps. In addition to limiting the scope of equation (2.15) to limited complexity maps, we include three further simple restrictions, namely 1) *Redundancy*: If N_I and N_O are the number of inputs and outputs respectively then we require $N_I \gg N_O$, so that $P(x)$ can in principle vary significantly, 2) *Finite size*: We impose $N_O \gg 1$ to avoid finite size effects, and 3) *Nonlinearity*: We require the map f to be a nonlinear function, as linear transformations of the inputs

cannot show bias towards any outputs[1]. These four conditions are not so onerous, and we expect that many real-world maps will naturally satisfy them.

We are still left with the problem that $K(x)$ is formally uncomputable. Nevertheless, in a number of real-world settings, $K(x)$ has been approximated using complexity measures based on standard lossless compression algorithms with surprising success [148]. That is to say, the approximations behave in a manner expected of the true Kolmogorov complexity, and lead to verified predictions. Example settings include: DNA and phylogeny studies [195, 79, 147], plagiarism detection [43], clustering music [49], and financial market analysis [247]; see Vitányi [230] for a recent review. These successes suggest that $K(x)$ can be approximated in some contexts [3], even if its exact value cannot be calculated. Following the successful applications of AIT above, we assume that the true Kolmogorov complexity $K(x)$ can be approximated by some standard method such as the ones described in the references above. We will call such an approximation the *approximate complexity* $\tilde{K}(x)$ to distinguish it from the true Kolmogorov complexity.

We therefore need a final condition 4) *Well behaved*: the map is 'well behaved' in the sense of not producing, for example, a large fraction of pseudorandom outputs. These non-well behaved maps would include maps produced by deterministic chaotic dynamical systems, as well as pseudorandom number generators, where similar input strings typically produce very dissimilar (and random-looking) outputs. An example of a pseudorandom output would be the digits of π, which are algorithmically simple, but which have large entropy and thus are likely to produce large values for the approximate complexity $\tilde{K}(x)$. In principle, if the method used to estimate $\tilde{K}(x)$ were able to identify pseudorandom outputs as simple (i.e. as low $\tilde{K}(x)$), this final condition would not be necessary.

Finally, the presence of $\mathcal{O}(1)$ terms in AIT expressions is perhaps the least understood limitation for applying formal AIT to real-world settings. Nevertheless, important recent work applying the full AIT coding theorem to very short strings [60, 212] has suggested that the presence of $\mathcal{O}(1)$ terms does not preclude the possibility of making decent predictions for smaller systems.

[1]The reason for this is because a uniform distribution of points in a given space, after a linear transformation, will remain a uniform distribution [26]. In particular, a uniform sample of points in input space, after a linear transformation, will result in a uniform distribution in output space, with no point in output corresponding to more inputs than any other point. Therefore a linear transformation cannot show any kind of bias.

Taken together, the arguments above allow us to make our central *ansatz*, namely that for many real-world maps, the upper bound in equation (2.11) can be approximated as:

$$P(x) \lesssim 2^{-a\tilde{K}(x)-b} \tag{2.18}$$

where the constants $a > 0$ and b depend on the details of the map, but not on x. These constants account for the $\mathcal{O}(1)$ terms and the particularities of the complexity approximation $\tilde{K}(x)$. Just as for the full coding theorem, there is a strong exponential decay in the probability upper bound upon a linear increase in complexity. This means that high probability outputs must be simple (have low $\tilde{K}(x)$), while high complexity outputs must be exponentially less probable. We call such phenomena that arise from equation (2.18) *simplicity bias*. We also show that the magnitude of the slope a can be approximated as

$$a \approx \frac{\log_2(N_O)}{\max\limits_{x \in O}(\tilde{K}(x))}. \tag{2.19}$$

where $N_O = |O|$ again denotes the number of outputs of the map. This connection between the slope a and N_O implies that one can be used to predict the other, if an estimate of $\max(\tilde{K}(x))$ can be found, which for some maps can be achieved quite easily. We also confirm a prediction made in ref. [66] that the *a priori* prediction for b is $b \approx 0$. Alternatively, a relatively small amount of sampled data is usually sufficient to fix b.

We examined the prediction of simplicity bias in a selection of input-output maps, including stochastic financial trading models, systems of coupled ordinary differential equations, as well as the RNA secondary structure map mentioned above. Note that the last one is also a GP map – after all, genotype-phenotype maps are a subset of the set of all input-output maps. This connection also suggests that perhaps simplicity bias should be listed along with other GP map properties discussed in Chapter 1, such as robustness, evolvability and shape-space covering.

2.6.1 Discrete RNA sequence to structure mapping

One of the best studied discrete input-output maps in biophysics is the RNA map discussed in Chapter 1. This map has as inputs RNA nucleotide sequences (genotypes), made from an alphabet of four different nucleotides, and as outputs the RNA secondary structures (SS), phenotypes which specify the bonding pattern of nucleotides [208]. This map is determined by basic physicochemical laws, and does not grow with n. Furthermore, the number of

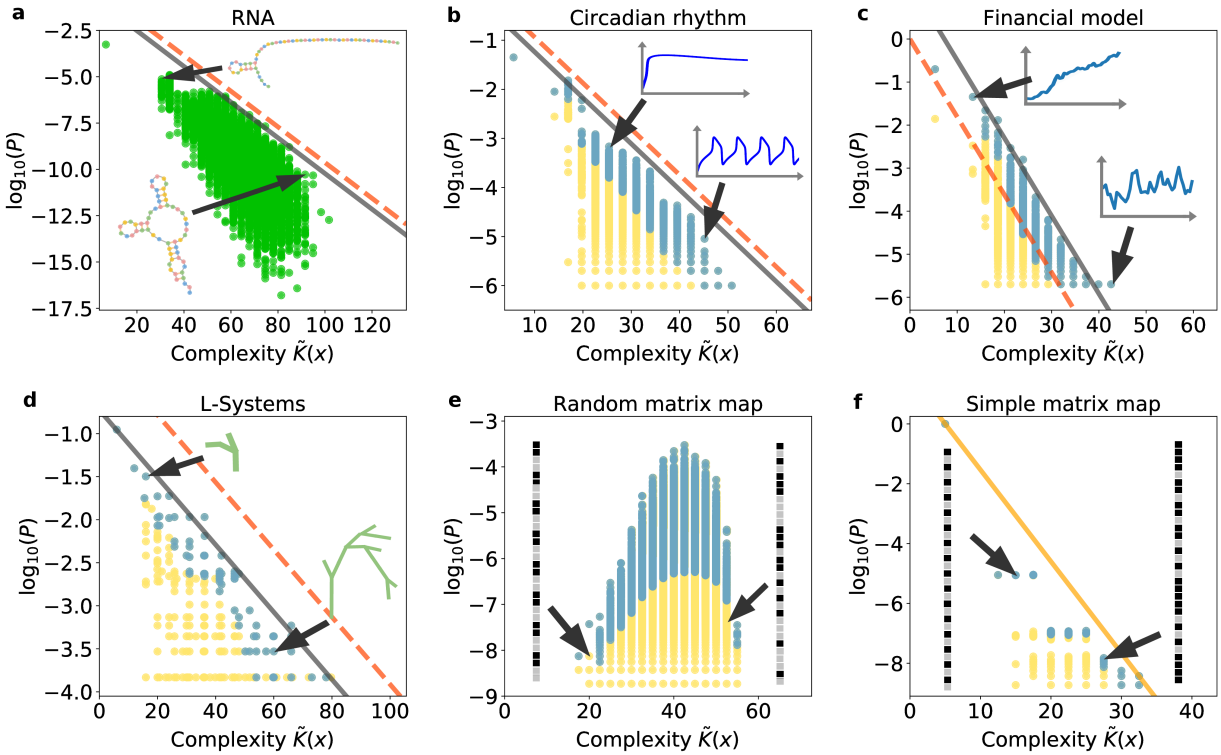


Figure 2.2: **Simplicity bias:** The probability $P(x)$ that an output x is generated by random inputs versus the approximate complexity $\tilde{K}(x)$ for different input-output maps: **(a)** the discrete $n = 55$ RNA sequence to SS map (less than 0.1% of outputs take up 50% of the inputs [68]) **(b)** the coarse-grained circadian rhythm ODE map (2% of the outputs take up 50% of the inputs), **(c)** the Ornstein-Uhlenbeck financial model (0.6% of the outputs take up 50% of the inputs), **(d)** L-Systems for plant morphology (3% of the outputs take up 50% of the inputs), **(e)** a random 32×32 matrix map and **(f)** a limited complexity 32×32 matrix map (both with less than 0.1% of the outputs taking over 50% of the inputs). Schematic examples of low and high complexity outputs are also shown for each map. Blue dots are probabilities that take the top 50% of the probability weight for each complexity value while yellow dots denote the bottom 50% of the probability weight (green was only used for **(a)**, the RNA map, because the output probabilities were calculated using the probability estimator described in ref. [112]). The bold grey lines denote the upper bound described in equation (2.18), while the dashed red lines represent the same upper bound, but with the default $b = 0$. For **(f)**, the upper bound line (orange) was fit to the distribution. All maps except for the random matrix map are of limited complexity, and exhibit simplicity bias.

inputs (sequences), which grows as 4^n, is much larger than the number of relevant secondary structures [208, 68], and so $N_I \gg N_O$. For large enough n, where finite size effects are no longer important [67], this system satisfies our conditions for simplicity bias.

In our analysis, 20000 random RNA sequences were generated, then folded to secondary structures using the Vienna package [106] with all parameters set to their default values (e.g. the temperature $T = 37°C$). Due to the large size of this system ($4^{55} \approx 1.3 \times 10^{33}$ inputs and approximately 10^{13} outputs), it is impractical to determine probabilities by sampling and counting frequencies of output occurrence. Instead, to determine $P(x)$ for each sampled structure, we used the neutral network size estimator (NNSE) described in ref. [112] which employs sampling techniques together with the inverse fold algorithm from the Vienna package. We used default settings except for the total number of measurements, which we set to 1 instead of the default 10, for the sake of speed. We used the same methods as in ref. [68], where the authors also calculate that only 0.1% of outputs take up over 50% of inputs, making this map highly biased.

To estimate the complexity of an RNA SS, we first converted the dot-bracket representation of the secondary structure provided by the Vienna package into a binary string x, and then used $C_{LZ}(x)$ to estimate its complexity. To convert dot-bracket strings to binary strings, we replaced each dot with `00`, each left-bracket with `10`, and each right-bracket with `01`. After this translation, an RNA SS of length n becomes a bitstring of length $2n$. As an example, Figure 2.3 shows how an $n = 15$ secondary structure results in a 30-bit string.

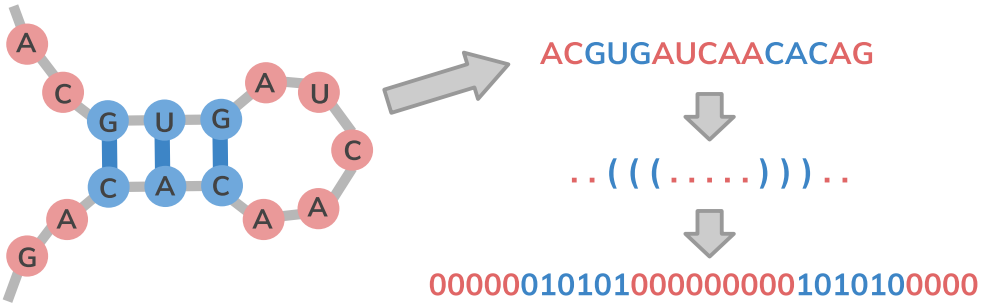


Figure 2.3: Example of the mapping from a $n = 15$ RNA structure to a dot-bracket string, which translates to a 30-bit string.

In Figure 2.2a we show $P(x)$ versus $\tilde{K}(x)$ for the $n = 55$ RNA map, with most inputs mapping to outputs relatively close to the upper bound. The slope $a = 0.32$ in our upper bound was estimated via equation (2.19) by using estimated values of N_O from ref. [68],

in addition to an estimated value for $\max(\tilde{K})$. For the latter quantity, we made the approximation that $\max(\tilde{K}) = C_{LZ}(\zeta_{2n})$, where ζ_{2n} is a random bitstring of length $2n$ made up of randomly choosing n pairs from $\{\texttt{00}, \texttt{10}, \texttt{01}\}$. We took the largest complexity over 10^4 random ζ_{2n} strings. This choice of randomisation is due to observing that a first order approximation to a random RNA structure is a random string composed of dots and pairs of brackets. Note that well-formed dot-bracket strings are in fact more restricted than this (for example, all brackets must open and close), and actual least energy RNA configurations may be even more restricted. That said, both ensembles of well-formed dot-bracket strings and random dot-bracket strings yield the same $\max(\tilde{K}) = 135.63$ bits. Finally, we estimate $b = 3.2$ using the sampled data to find the maximum probability within outputs with complexity equal to the modal $\tilde{K}(x)$.

2.6.2 Coarse-grained ordinary differential equation

Ordinary differential equation (ODE) models can be coarse-grained into discrete maps by discretising both the input parameters and the outputs. As an example, we take a well studied model for the circadian rhythm of eukaryotes [229], a system of 9 nonlinear ODEs:

$$\begin{aligned}
dD_A/dt &= \theta_A D'_A - \gamma_A D_A A \\
dD_R/dt &= \theta_R D'_R - \gamma_R D_R A \\
dD'_A/dt &= \gamma_A D_A A - \theta_A D'_A \\
dD'_R/dt &= \gamma_R D_R A - \theta_R D'_R \\
dM_A/dt &= \alpha'_A D'_A + \alpha_A D_A - \delta_{M_A} M_A \\
dA/dt &= \beta_A M_A + \theta_A D'_A + \theta_R D'_R - A(\gamma_A D_A + \gamma_R D_R + \gamma_C R + \delta A) \\
dM_R/dt &= \alpha'_R D'_R + \alpha_R D_R - \delta_{M_R} M_R \\
dR/dt &= \beta_R M_R - \gamma_C A R + \delta_A C - \delta_R R \\
dC/dt &= \gamma_C A R - \delta_A C
\end{aligned} \tag{2.20}$$

Since we are mainly using this model to illustrate a generic ODE map, we will not give a complete description of what all the parameters mean. For a full account we direct the reader to the original paper [229]. Very briefly, the model above aims to study the ability of circadian clocks to maintain a constant period even in noisy conditions, and describes the interaction of two genes that regulate the expression of a pair of proteins, the activator A and

the repressor R. This repressor works by sequestering the activator, forming the inactivated complex C, whose concentration over time is the variable we chose to use as output, taking its rate of formation (i.e its slope) at discrete time steps as the output of the map. Since this variable is placed at the bottom of the regulatory cascade, it is a natural choice for the output. Figure 2.4 represents the output of the ODE model, showing the concentration of the nine molecules over time, for the set or parameters given in the original paper [229]: D_A and D_R start at 1 molecule, meaning one single copy of the activator and repressor genes, and other variables start at zero, with $\alpha_A = 50$ h^{-1}, $\alpha'_A = 500$ h^{-1}, $\alpha_R = 0.01$ h^{-1}, $\alpha'_R = 50$ h^{-1}, $\beta_A = 50$ h^{-1}, $\beta_R = 5$ h^{-1}, $\delta_{MA} = 10$ h^{-1}, $\delta_{MR} = 0.5$ h^{-1}, $\delta_A = 1$ h^{-1}, $\delta_R = 0.2$ h^{-1}, $\gamma_A = 1$ molecule^{-1} h^{-1}, $\gamma_R = 1$ molecule$^{-1}h^{-1}$, $\gamma_C = 2$ molecule^{-1} h^{-1}, $\theta_A = 50$ h^{-1}, and $\theta_R = 100$ h^{-1}.

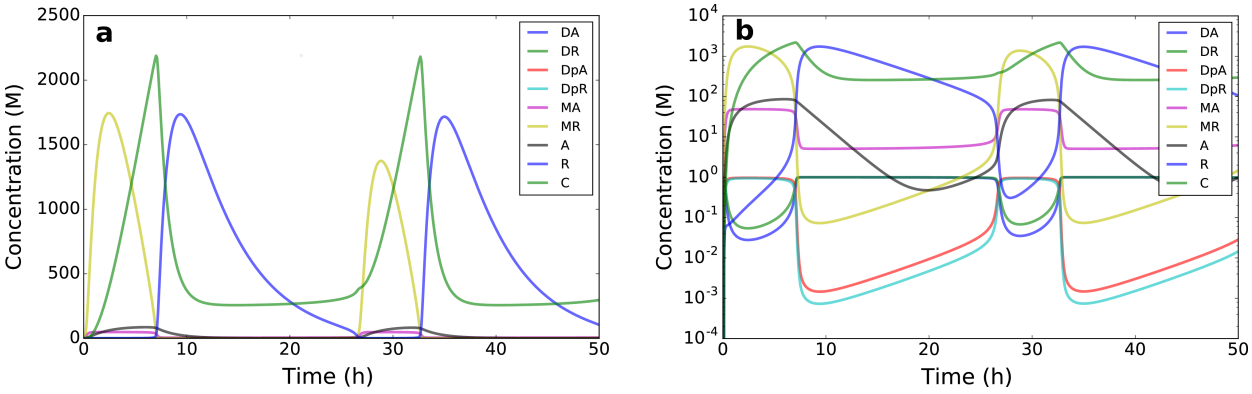


Figure 2.4: The output of the ODE model described in equation (2.21), showing the concentration of 9 molecules from equation (2.20) over time, in linear scale **(a)** and logscale **(b)**. The initial conditions and parameters are the same as in the original paper [229].

Since in this input-output map the inputs are given as continuous parameters, sampling inputs is not as simple as in the RNA map, where the inputs are discrete strings. Here, instead, input parameters correspond to allosteric constants and affinity rates, and the realistic ranges for such parameters are often unknown. Moreover, the value of each parameter is also the product of a series of other very complex input-output maps, as they are the result of information passed from DNA sequences into amino acid sequences and eventually into protein function. To make progress, we set all 15 parameters to their wild-type values, multiplied by a random factor in $\{0.25, 0.50, \ldots, 1.75, 2.00\}$ chosen with uniform probability of $\frac{1}{8}$. This approach is inspired by the robustness sampling in Chen et al's model of the budding yeast [42], where the authors also multiply the wild-type parameters by a range of values. In this way we effectively have a discrete set of input parameters.

Since the output variables of this map depend continuously on the input variables, it is also necessary to coarse-grain outputs into discrete values. In principle, every input would produce a unique output; however, many of those outputs can be very similar to one another. To capture this simililarity, we coarse-grain the outputs by discretising them into binary strings using the "up-down" method, by Fink et al [80]. We take the output curve $y(t)$ calculated in an interval $t \in [0, T]$, calculate its slope dy/dt at intervals of $t = \delta t, 2\delta t, 3\delta t, \ldots$, and print the sign of dy/dt in every interval: for $j = 1, \ldots, T/\delta t$, if $dy/dt \geq 0$ (or < 0) at $t = j\delta t$, the j-th bit of the output string gets assigned a `1` (or a `0`). The resulting string represents the oscillations of $y(t)$: curves with more oscillations will produce more complex strings, while curves with fewer oscillations will produce strings with longer repeated sequences of `0`s and `1`s. The complexity of this mapping procedure does not depend on n, and so the whole map from parameters of the ODE model to binary output strings can be viewed as a limited complexity map.

It is important to notice that simplifying the output of the ODE model to a binary string does not introduce simplicity bias. To explain that, consider an input-output map describing a one-dimensional Brownian motion, taking random seeds as input and producing the position of the random walkers over time as output. Given that a Brownian motion is symmetric regarding the origin, it is equally likely for the random walker to go up or down at any given moment, implying that this map does not have simplicity bias – in fact, it is not biased towards any particular trajectory. This symmetry around the origin also implies that, if one were to apply Fink et al.'s "up-down" method to the output trajectories produced by this map, the resulting binary strings would be equally likely to have `1`s or `0`s at any position of the string. The binary outputs would be overall simpler than the continuous trajectories produced by the Brownian motion, but the map would still show no simplicity bias, as all outputs would still be equally probable.

To generate the plot in Figure 2.2b, we took a sample of 10^6 inputs, and outputs were discretised with $\delta t = 1$ for 50 time steps, producing a 50-bit output bitstring. The slope $a = 0.31$ was obtained via equation 2.19, using the values of N_O and $\max(\tilde{K})$ from the full enumeration of inputs, while $b = 1.0$ was fit to the distribution.

As seen in Figure 2.2b, the probability $P(x)$ decays strongly as the approximate complexity of the outputs increases. This coarse-grained ODE map shows the same broad simplicity bias behaviour as the RNA map. Again the upper bound works remarkably well, given the

relatively small amount of information needed about the map to fix it. The majority of points generated by random sampling of inputs are within one or two orders of magnitude from the bound.

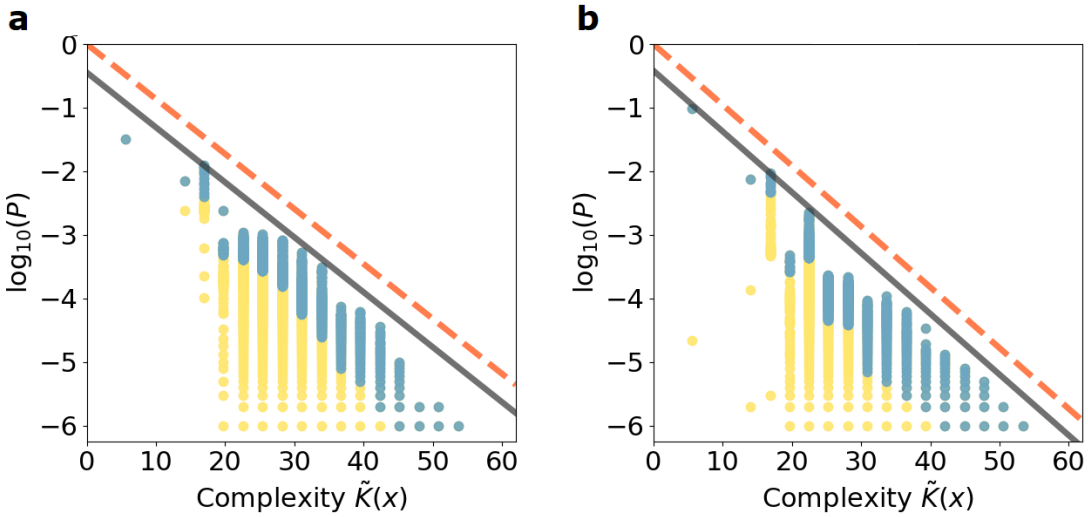


Figure 2.5: **Probability versus complexity for the outputs of a circadian rhythm model.** Subfigure **(a)** shows the result of 10^6 inputs sampled uniformly from a continuous range, instead of sampled from discretised values over the same range. The upper bound coefficients are $a = 0.29$ and $b = 1.48$. Subfigure **(b)** shows the result of 10^6 combinations of inputs and initial conditions, sampled from a discretised range. The upper bound coefficients are $a = 0.32$ and $b = 1.39$. For both plots, outputs were discretised in 50 bins, and the upper bound coefficients a and b were derived according to equation (2.19). The results are very close to Figure 2.2. For every value of $\tilde{K}$, the blue dots represent the outputs that correspond to the top 50% of the probability weight of the inputs that produce outputs with $\tilde{K}(x) = \tilde{K}$.

Alternative discretisation of inputs

To confirm that our discretisation method does not affect the simplicity bias we observe, we produced a sample of 10^6 inputs taken uniformly on the whole of our estimate of the parameter range for each parameter, rather than taken from a discrete set of parameter values. The result is shown in Fig 2.5a: we obtain the same simplicity bias as observed in Figure 2.2, with the values of a and b within the uncertainty of our methods for obtaining them.

Discretising initial conditions

Since the behaviour described by the circadian rhythm model should depend on the initial conditions of the ODE system, it is important to test whether the simplicity bias we observe

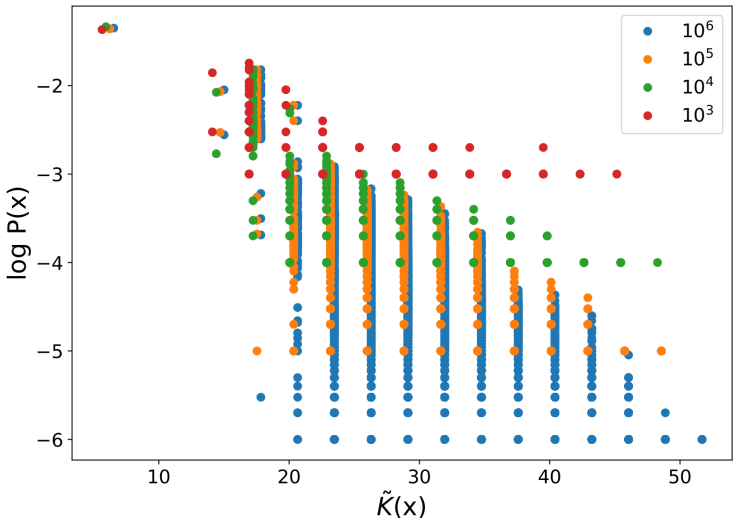


Figure 2.6: **Probability versus complexity for the outputs of a circadian rhythm model with 10^3 to 10^6 inputs, and outputs discretised in 50 bins.** Every sample size is displaced horizontally from the previous one by 0.3 for clarity. For a sample of 10^n inputs, the lowest possible probability is 10^{-n}. These are outputs that only appear once in the sample, resulting in an inaccurate estimate of their probability, which becomes more accurate as the size of the input sample grows. On the other hand, the $P(x)$ close to the upper bound do not change much with as sample size grows, which results in the upper bound remaining roughly the same for different sample sizes.

also depends on that. We took a sample of 10^6 combinations of different values for parameters and initial conditions, produced their corresponding outputs and derived the upper bound coefficients $a = 0.32$ and $b = 1.39$ according to equation (2.19). The resulting map is plotted in Figure 2.5b, which show the same simplicity bias behaviour as seen in Figure 2.2.

Alternative input sample sizes

We also confirm that the choice of sample size also does not affect our results in any qualitative way. To generate the plot shown in Figure 2.2b, we took a sample of 10^6 inputs, which is far less than total $8^{15} \approx 3 \times 10^{13}$ inputs for this coarse-grained ODE model. Any estimate of $P(x)$ for outputs for which $P(x) \lesssim 10^{-6}$ will be subject to large errors. We also tested different sample sizes, ranging from 10^3 to 10^6, as seen in Figure 2.6, and the smaller input samples follow an upper bound with a slope very close to the slope found for the larger input samples. This demonstrates that the upper bound slope can be estimated without a full enumeration of the inputs, even in the situations where very small sample sizes do not provide a good estimate of N_O, which would normally be used by equation (2.19) to estimate the slope a.

Multiple levels of coarse-graining the outputs

Figure 2.7 shows the result of varying the number of bins for Vilar et al's circadian rhythm model [229]. Naturally, having more bins results in a wider range of values for the phenotype complexity (and larger values overall), but the plots for different numbers of bins all show the same result: phenotype probability varies in orders of magnitude, obeying a roughly exponential bound on complexity $\tilde{K}(x)$, with the most frequent phenotypes being the lowest in complexity.

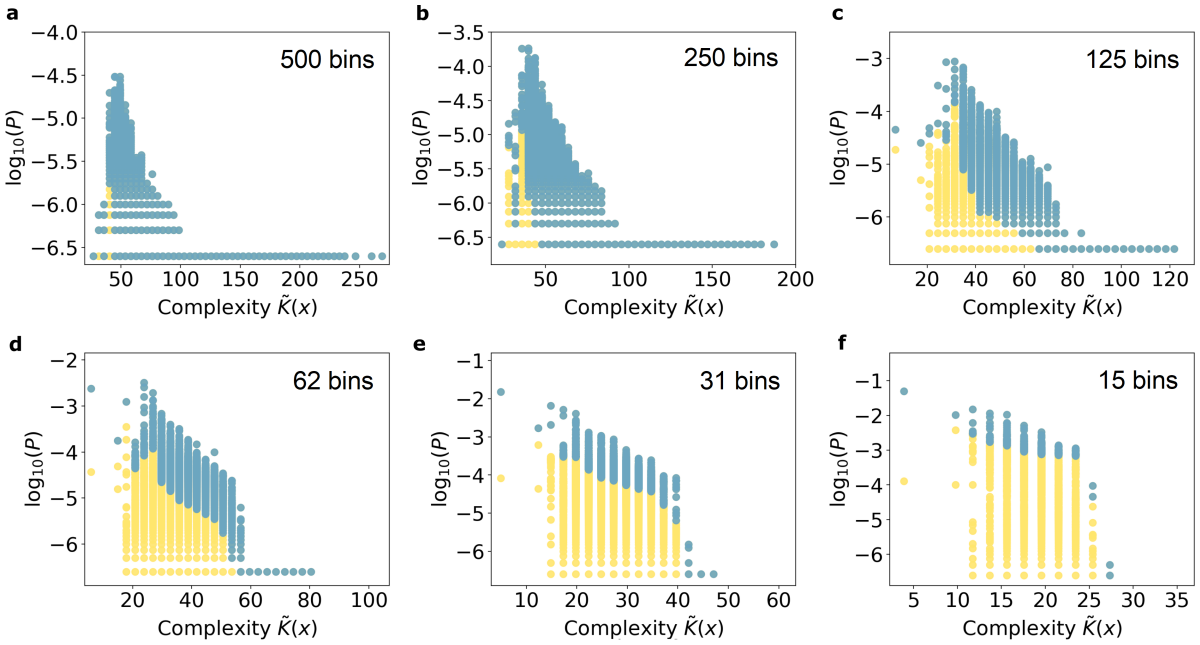


Figure 2.7: Different runs of the ODE model describing the circadian rhythm by Vilar et al. [229] for increasing levels of coarse-graining. Each subplot presents phenotype probability versus phenotype complexity for the same set of 5×10^6 genotypes, with the ODE output discretised in **(a)** 500 bins, **(b)** 250 bins, **(c)** 125 bins, **(d)** 62 bins, **(e)** 31 bins, **(f)** 15 bins. For every value of $\tilde{K}$, the blue dots represent the outputs corresponding to 50% of the inputs that produce outputs with $\tilde{K}(x) = \tilde{K}$. The decrease in the blue-shaded area as the number of bins increases indicates that the map is more biased, as fewer outputs are needed to cover 50% of genotype space.

2.6.3 Ornstein-Uhlenbeck stochastic financial trading model

In mathematical finance, the Ornstein-Uhlenbeck process, or Vasicek model [28], is used to model interest rates, currency exchange rates, and commodity prices, and is applied in a trading strategy known as pairs trade [141]. This stochastic process is governed by $dS_t = \theta(\mu - S_t)dt + \sigma dW_t$, where μ represents the historical mean value, σ denotes the

degree of market volatility, θ is the rate at which noise dissipates, and W_t is a Brownian motion, which is taken as the input to this map. These inputs then generate S_t, and the outputs x_t are defined as sequences over n time steps, where $x_j = 0$ if $S_j \leq 0$, and $x_j = 1$ otherwise. The output sequence x can be interpreted as indicating whether S_t is above or below its historical average, and thus whether the trader would profit by selling or buying more of it. We measure the complexity of these binary output strings using $C_{LZ}(x)$, as for the other maps in this chapter. As shown in Figure 2.2c, this map shows basic simplicity bias phenomenology, and the prediction of upper bound slope a based on equation (2.19) also works well.

For the Ornstein-Uhlenbeck model presented in Figure 2.2c, the parameters were $S_0 = 1$, $\theta = 0.5$, $\mu = 0.5$ and $\mu = 0$, and 40 time steps. 10^6 samples were made, and given that the Brownian motion dW_t allows for steps of any size (even at a low probability), the space of all possible outputs O comprises all 2^{40} binary strings of length 40, making it possible to calculate $\max(\tilde{K}(x))$ and N_O. The slope $a = 0.60$ was obtained via equation 2.19, and $b = -4.38$ was obtained using the modal complexity value. In Figure 2.8 we confirm that this bias towards simplicity is not an artefact of our choice of parameters, as different values of θ and σ still produce bias towards simple behaviour.

2.6.4 L-systems for plant morphology

Lindenmayer systems or L-systems [152] are a general modelling framework originally introduced for modelling plant growth, but now also used extensively in computer graphics. They consist of a string of different symbols which constitute production rules for generating geometrical shapes. We confined our investigation to non-cyclical graph (i.e. topological tree) outputs. An example of L-system is shown in Figure 2.9. The rule set defining the L-systems is independent of input length, making this a limited complexity map.

For the plot in Figure 2.2d, we enumerated all valid L-systems consisting of a single starting letter F, followed by rules made of symbols from $\{+, -, F\}$ and depth ≤ 9. The outputs were coarse grained using a method suggested by ref. [167] to associate binary strings to any non-cyclical graph with a distinguished node called the 'root' (these graphs are known as *rooted trees*), by performing a depth-first search: starting from the root, one walks along the branches, starting right, and recording whether it is going up (`0`) or down (`1`), thus producing a binary string representation of the tree. Simplicity bias is also present

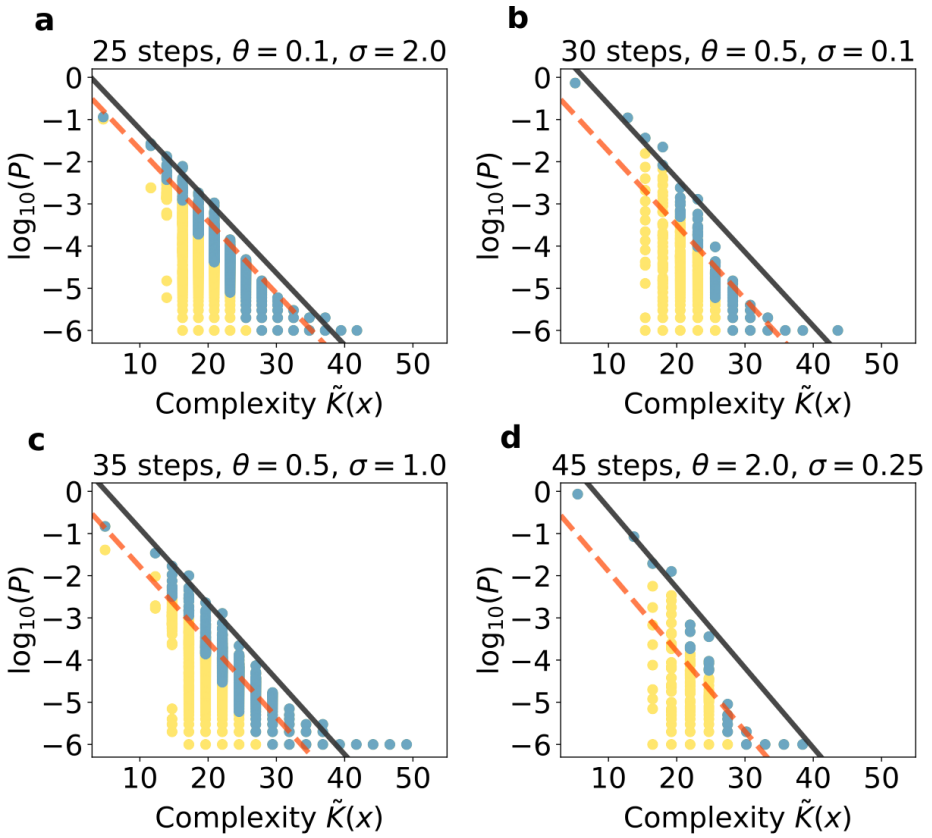


Figure 2.8: **Probability versus complexity for the outputs of a Ornstein-Uhlenbeck for different choices of the parameters θ and σ, as well as of the number of steps of the Brownian motion.** Respectively: **(a)** $\mu = 0.1$, $\sigma = 2.0$, 25 steps, a=0.56, b=-1.6, **(b)** $\mu = 0.5$, $\sigma = 0.1$, 30 steps, a=0.58, b=-3.7, **(c)** $\mu = 0.5$, $\sigma = 0.10$, 35 steps, a=0.59, b=-3.0, **(d)** $\mu = 2.0$, $\sigma = 0.25$, 45 steps, a=0.63, b=-5.0. For every value of $\tilde{K}$, the blue dots represent the outputs corresponding to 50% of the inputs that produce outputs with $\tilde{K}(x) = \tilde{K}$. Since $N_O = 2^n$, $\max(\tilde{K}(x))$ is also known, and the upper bound coefficients were calculated using equation (2.19). Red dashed lines represent an upper bound offset of $b = 0$.

in this system, and the slope $a = 0.13$ was obtained via equation 2.19, using the values of N_O and $\max(\tilde{K})$ from the full enumeration of inputs, while $b = 2.41$ was obtained using the modal complexity value, as for other maps above.

2.7 The matrix map

Finally, we provide an example of a map that exhibits strong bias but not necessarily simplicity bias. We define the *matrix map* as an input-output map made of binary input vectors p of length n that map to binary output vectors x of same length through $x_i = \Theta((M \cdot p)_i)$, where M is a $n \times n$ matrix and the Heaviside thresholding function $\Theta(y) = 0$ if $y < 0$ and $\Theta(y) = 1$ if $y \geq 0$. The threshold function ensures the nonlinearity of this

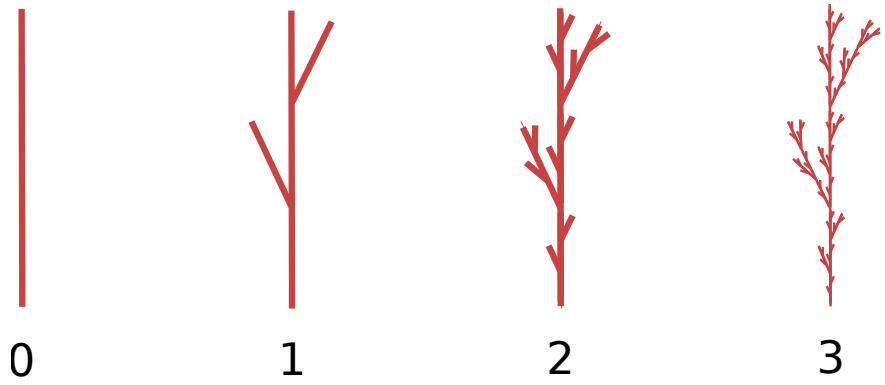


Figure 2.9: Example of the production steps for a L-system of depth 3, produced by applying the rule `F = F+[F]-F-[F]+F` over three iterations.

input-output map, since a linear map such as $z = M \cdot p$, is incapable of producing bias towards any output, as discussed in Chapter 2. The map is illustrated in Figure 2.10.

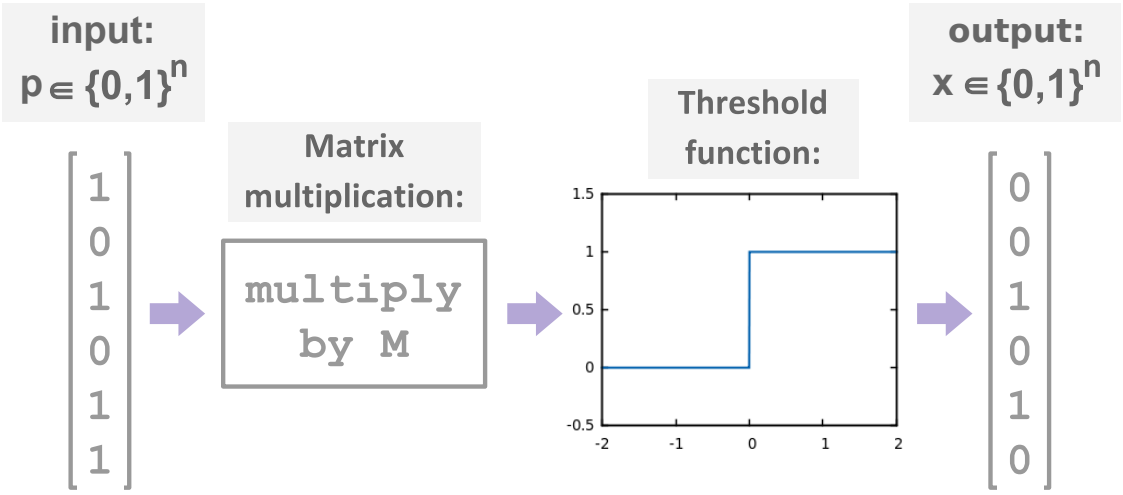


Figure 2.10: Illustration of an input-output map consisting in binary vectors multiplied by a matrix, following by a binary threshold so that the output also consists in binary vectors.

The matrix map can also be seen as a simple neural network made only by the input and output layers, each with n units, having a step function as its activation function, and no hidden layers. This perspective will be explored in Chapter 6. In this section, we treat this map as a very simple and transparent way to map a set of inputs to their corresponding outputs, through a matrix transformation. The matrix map is also *tunable*, in the sense that using different matrices M will result in maps of different complexity.

2.7.1 Random matrix maps: bias without simplicity bias

Figure 2.11a shows an example of a matrix map, produced from a random 20×20 matrix where every entry is chosen from $\{-1, 1\}$ with uniform probability. The figure shows that this map generates a very biased distribution of inputs over outputs, but, in sharp contrast to the other systems studied in this paper, there is no bias towards simpler or more complex outputs, as it can be seen in Figure 2.11b.

Figure 2.2e also shows output probability versus complexity for a random 32×32 matrix map where the matrix entries are randomly chosen from $\{-1, +1\}$. The figure shows the results for a sample of 10^9 inputs. Rather than the triangular shape seen in other plots of probability versus complexity, Figure 2.11b and Figure 2.2e show a parabolic upper bound – or a bell shape, if one were to draw the y axis in linear scale. This distribution of output complexity is discussed in Appendix A.

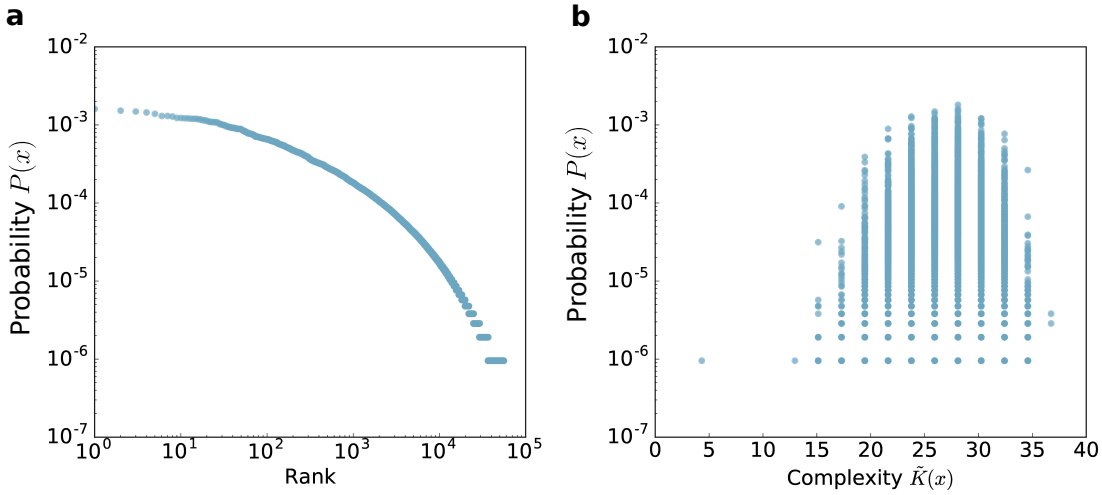


Figure 2.11: Properties of a matrix map made from a random 20×20 matrix where every entry is chosen from $\{-1, 1\}$ with uniform probability. The plots show the results corresponding to all 2^{20} inputs. **(a)** A rank plot indicates that this map has an uneven distribution of inputs over outputs. **(b)** In contrast to simplicity bias phenomenology, there is no clear correlation between output probability and output complexity.

The map shown in Figure 2.11 is not an isolated example: in fact, most matrix maps in our samples produce a biased distribution of inputs over outputs. We can verify this using a very simple measure of bias first introduced in ref. [68]. If one takes the entropy of the distribution of the probabilities p_i of the N_O outputs, $H = -\sum_{i=1}^{N_O} p_i \log_2 p_i$, then the exponential of the entropy, 2^H, should be a rough estimate of the effective number of outputs [171]. One can then define a *bias ratio* $\beta = 2^H/N_O$, which measures the ratio

between the effective number of outputs and the total number of outputs in that map. For maps with $\beta \approx 1$, inputs should be roughly uniformly distributed across outputs, while for maps with $\beta \ll 1$, most inputs should correspond to a few outputs. In other words, the closer β is to 0, the stronger the bias in the distribution of inputs over outputs. For example, the map in Figure 2.11 has a bias parameter $\beta = 0.56$. In Figure 2.12a we show a histogram produced from an ensemble of 5000 matrices where each of the 20×20 entries was chosen uniformly from $\{-1, 1\}$. We observe that most matrix maps in that ensemble have small β, and so most inputs map to a small fraction of the outputs. In other words, the maps show bias.

For this same set of 5000 matrices we calculated the probability of a given output, and plotted it versus its complexity. The resulting distribution, shown in Figure 2.12b, is very close to what we would expect for a random sample of binary strings of length 20. We also sampled over 5000 random 20×20 matrices where every entry was taken from a standard normal distribution ($\mu = 0, \sigma = 1$), finding very similar results.

In addition to showing results for a set of uniformly sampled outputs (indicated as "o-sampling" in the figure), Figure 2.12b also shows the probability versus complexity for a set of outputs produced by uniform sampling of inputs, which are then fed through the map ("i-sampling" in the figure). The comparison between these two kinds of sampling is important when looking at input-output map properties such as the ones discussed in Chapters 1 and 2, such as bias and simplicity bias: for instance, if taking a random input and passing it through the map is likely to produce a relatively simple output, the map shows signs of simplicity bias.

Even though every matrix gives rise to a unique input-output map, when we subsequently average over the matrices, the difference between them goes away. For the matrices with randomly chosen $\{-1, 1\}$ entries, there remains a very slight difference between input and output sampling, while for the matrices with entries taken from a standard normal distribution input and output sampling yield essentially the same results within our measurement errors. In other words, while for a single matrix with bias, certain outputs are much more likely to be generated by random sampling of inputs than others, when averaged over many matrices, no output string is more or less likely to be generated than any other. This behaviour is fundamentally different from maps that show simplicity bias, because even if we

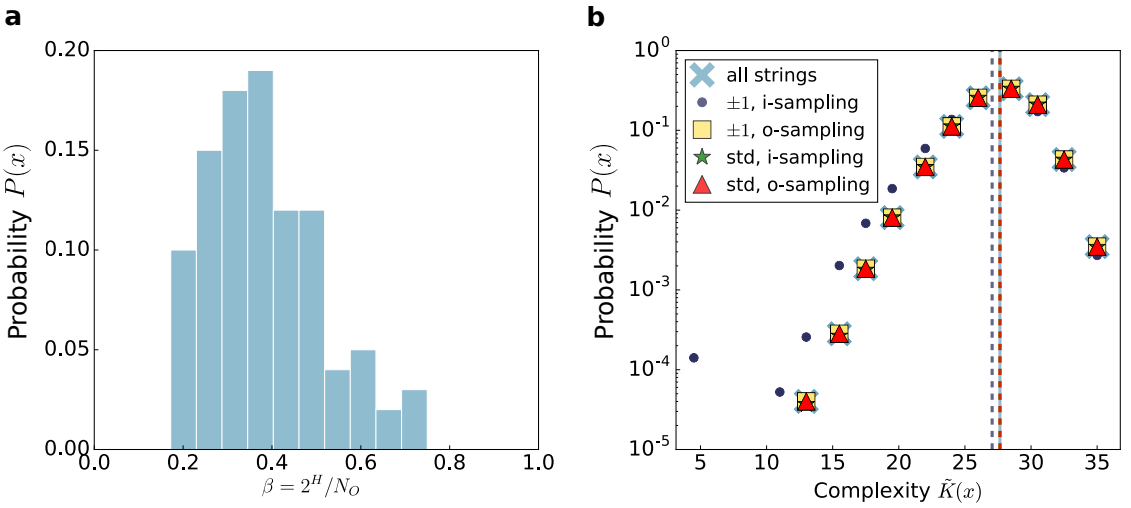


Figure 2.12: **Distribution of outputs for random 20×20 matrix maps.** The plots show the results corresponding to all 2^{20} inputs. **(a)** Histogram for the bias ratio β, a measure for the ratio between the number of effective outputs and the total number of outputs of a map. Smaller values of β mean more bias in the map. The map in Figure 2.11 has a bias parameter $\beta = 0.56$. **(b)** Distribution of $\tilde{K}(x)$ for the outputs of matrix maps from an ensemble of random 20×20 matrices with entries chosen uniformly from $\{-1, 1\}$. Dark blue circles denote a distribution made by sampling inputs and yellow squares denote a distribution made by uniformly sampling over outputs, The green stars and red triangles respectively represent the same input-sampled and output-sampled averages, but for random 20×20 matrices with taken from a standard normal distribution ($\mu = 0, \sigma = 1$). The light blue crosses represent the distribution of $\tilde{K}(x)$ over all bitstrings of length 20. The overlap between all curves shows that the outputs of a random matrix map are likely to be as complex as a random set of strings of the same length. Error bars are too small to be visible, and average values are all within 27.3 ± 0.3.

averaged over maps, we would expect low complexity outputs to be more likely to occur on average.

We argue that the reason simplicity bias does not occur for the random matrix map is because this map violates the limited complexity condition described in subsection 2.6. Having to define $n \times n$ independent matrix elements means that the complexity of the map grows as $\mathcal{O}(n^2)$, which places it outside of the range of limited complexity maps for which we predict simplicity bias. Still, the random matrix map shows that an input-output map can show bias without showing simplicity bias.

2.7.2 Circulant matrices can show simplicity bias

All the matrix maps considered so far were made from random matrices, with $n \times n$ entries either taken from $\{-1, 1\}$ or from a normal distribution of mean zero and variance 1. In other words, for square matrices, the amount of information needed to specify the map

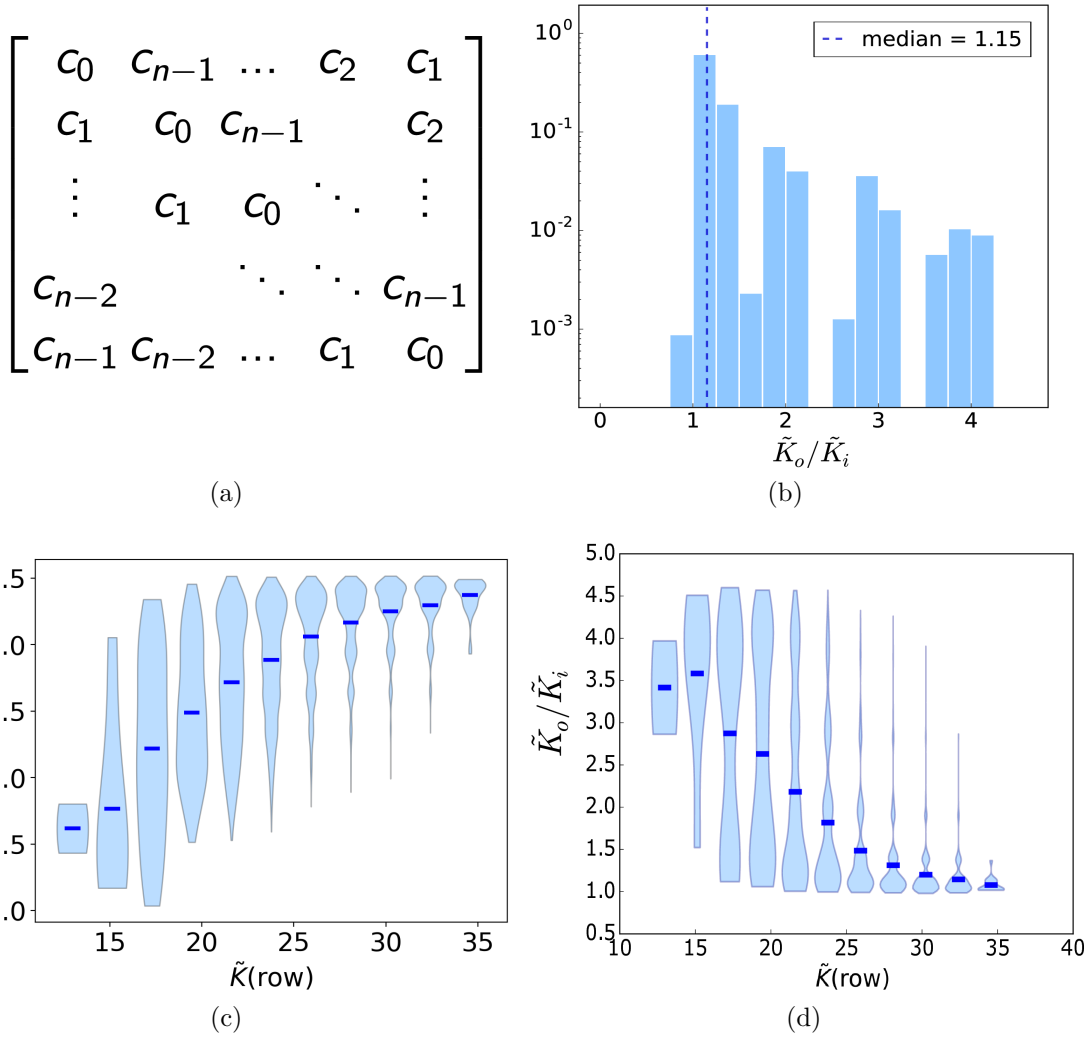
grows as $\mathcal{O}(n^2)$. In this section we study a set of matrix maps for which the amount of information needed to specify the map grows more slowly with n.

We begin by studying circulant matrices, illustrated in Figure 2.13a. Circulant matrices are a class of Toeplitz matrices where each row corresponds to the row above, shifted to the right by one element. They play a role in fields ranging from discrete Fourier transforms to cryptography [95]. In this work, their most important aspect is that these matrices are defined by the values on the first row. Limiting our matrix entries to $\{-1, 1\}$, a circulant matrix can be defined using n bits or less, as opposed to n^2 bits for a random $\{-1, 1\}$ matrix. These matrices are illustrated in Figure 2.13a.

While the outputs from the random matrix maps explored in the previous section were, on average, as complex as any random binary string of the same length, some circulant matrix maps do show some bias towards simple outputs (Some fully random maps may also show simplicity bias, but these are sufficiently rare that we did not find any in our sampling). Figure 2.13b shows the ratio between $\tilde{K}_o$, the average output complexity when all outputs are assigned the same weight, and $\tilde{K}_i$, the average output complexity where every output is weighted by the frequency with which it appears upon random sampling of inputs. The majority of maps have $\tilde{K}_o/\tilde{K}_i \approx 1$, but a small fraction (a few percent) have significantly higher ratios, which mean that low complexity outputs are more likely to appear upon random sampling of inputs.

Figures 2.13c and 2.13d show how $\tilde{K}_o$ and $\tilde{K}_o/\tilde{K}_i$ vary with the complexity of the first matrix row that defines the map. The distributions are shown in a standard violin plot format, and the horizontal dark blue lines denote the mean value on the y axis for each value of $\tilde{K}(row)$. Figure 2.13d indicates that only relatively simple matrix maps exhibit simplicity bias, as indicated by $\tilde{K}_o/\tilde{K}_i$ being significantly greater than 1, but Figure 2.13c shows that $\tilde{K}_o$ goes down with row complexity as well, suggesting that low complexity maps are likely to produce simpler outputs, regardless of how many inputs map to each output. This additional form of simplicity seems to be independent of simplicity bias and will be explored in Chapter 5.

Since random matrix maps violate the limited complexity assumption, one might wonder if it is possible to make maps that do not violate this rule. We can explore this behaviour in more detail by creating circulant matrices with low complexity rows. We limited the $\{-1, 1\}$ matrix ensemble to 20×20 circulant matrices defined by rows containing $\log n$ positive entries



(a) (b) (c) (d)

Figure 2.13: **Properties of circulant matrices.** **(a)** Illustration of a circulant matrix. **(b)** Histogram for the ratio $\tilde{K}_o/\tilde{K}_i$ in the matrix maps made from a sample of 2.5×10^4 circulant matrices with 20×20 entries taken from $\{-1, 1\}$. $\tilde{K}_o/\tilde{K}_i$ measures the ratio of the mean complexity of all individual outputs of a given map, divided by the mean complexity of outputs generated by random sampling over all inputs. **(c)** Violin plots showing how the average output complexity $\tilde{K}_o$ changes with the complexity of the top row of the same circulant matrices. **(d)** Violin plots showing how $\tilde{K}_o/\tilde{K}_i$ changes with the complexity of the top row of the same circulant matrices.

(and negative entries otherwise). Those matrices, by construction, are defined by $\mathcal{O}(\log_2 n)$ bits, a much smaller amount of information when compared to the $n \times n$ bits required to specify each entry of a random matrix with entries taken randomly from $\{-1, 1\}$, or the n bits to define most circulant matrices in the full ensemble, since most strings of length n have near maximum C_{LZ} complexity.

For the matrix maps made from these simpler matrices, the evidence for simplicity bias is clear: all maps show a ratio $\tilde{K}_o/\tilde{K}_i > 3.5$, as shown in Figure 2.14. Since those matrices will consist in 320 entries equal to -1 and 80 entries equal to $+1$, we ran a control using using matrices with these same proportions, but without the row-by-row structure, as well as matrices with the same proportion of $+1$ and -1 per row $(16 : 4)$, but without the Toeplitz structure. We tested both alternatives, but none showed simplicity bias as the simple matrices described in this section do. This can be attributed to the fact that these maps need not $\mathcal{O}(\log_2 n)$ bits to be described, but $\mathcal{O}(n \log n)$, since all row structure is lost. They thus violate our limited complexity condition that $K(f) + K(n) \ll K(x)$.

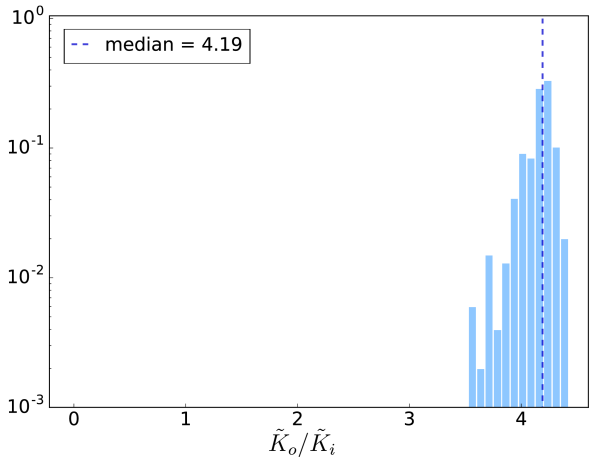


Figure 2.14: Histogram for $\tilde{K}_o/\tilde{K}_i$ for 20×20 circulant matrices with entries taken from $\{-1, 1\}$, containing only $\lfloor \log 20 \rfloor = 4$ positive entries per row. The outputs corresponding to all 2^{20} inputs produce a ratio of $\tilde{K}_o/\tilde{K}_i \approx 3.5$.

For the matrix map represented in Figure 2.2f, we sampled 10^9 inputs for a circulant 32×32 matrix made from a row of low complexity. In this map the estimate of the slope from equation (2.19) did not work well, possibly because a large fraction of the inputs map to a single output vector made up of all 0s. Due to that issue, in Figure 2.2f, the slope was fit to the data.

Matrix rank and sparsity do not explain simplicity bias

Matrix rank and sparsity are two traditional measures for the complexity of matrices. Since a typical random matrix has high rank, as shown in Figure 2.15, it could be possible that matrix maps using low-rank matrices would produce simplicity bias. In this section, we looked at $\tilde{K}_o$, $\tilde{K}_i$, and their ratio for 10×10 matrices of rank and sparsity varying from 1 to their maximum values of 10 and 100 respectively.

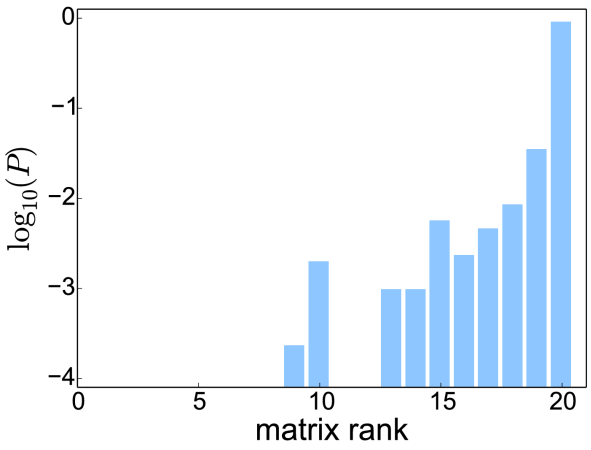
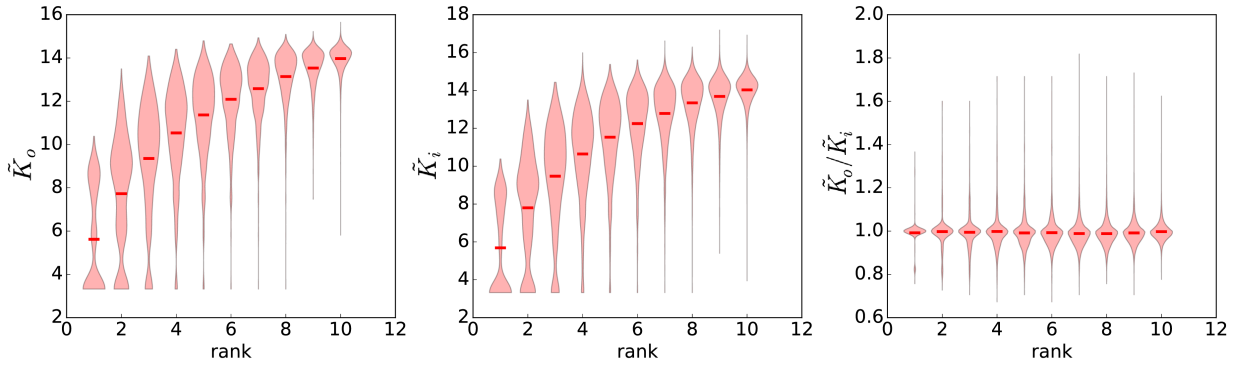


Figure 2.15: Histogram for the matrix rank of 25000 circulant 20×20 matrices with entries taken from $\{-1, 1\}$, for all 2^{20} inputs.

Firstly, in Figure 2.16 we show violin plots for the output complexity versus matrix rank, for random 10×10 matrix maps made from random $\{-1, 1\}$ matrices. As the matrix rank decreases from 10 to 1, both $\tilde{K}_o$ and $\tilde{K}_i$ deviate from the full-rank matrices. Figure 2.16c, however, shows that the ratio $\tilde{K}_o/\tilde{K}_i$ stays constant. In other words, even though matrix maps made from matrices of lower rank do produce low complexity outputs, that is not because those outputs correspond to more inputs than their high complexity counterparts, but rather because the high complexity outputs are not being produced at all.

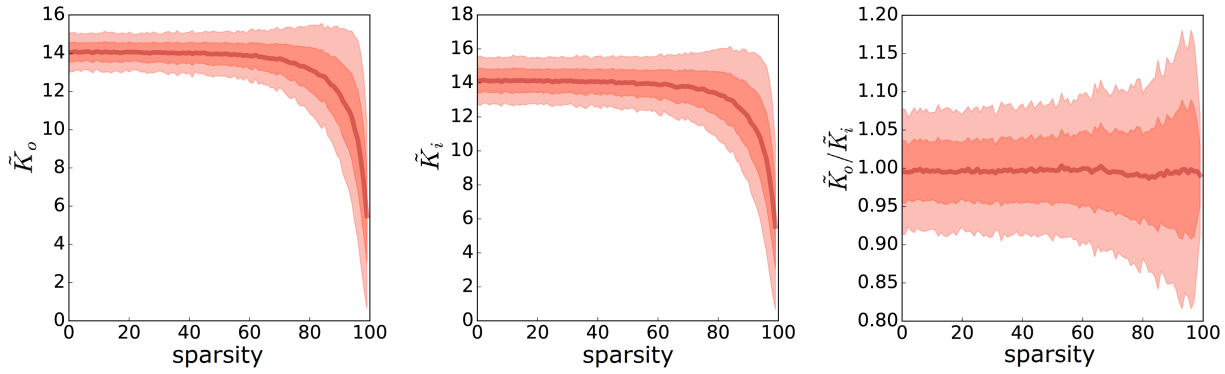
The second natural candidate for the complexity of a matrix map is the sparsity of its matrix, i.e. the number of zeros in the matrix. As shown in Figure 2.17 for 10×10 matrices, $\tilde{K}_o$ and $\tilde{K}_i$ remain the same for levels of sparsity up to 80%, and drop sharply for sparser matrices. Still, the ratio $\tilde{K}_o/\tilde{K}_i$, remains around 1, indicating once again a decrease in output complexity but no bias towards producing simple outputs.

In summary, even though rank and sparsity are used as proxies for matrix complexity in other contexts, we have presented evidence that they only affect the average output



(a) (b) (c)

Figure 2.16: **Violin plots for the complexity of random $\{-1, 1\}$ matrix maps made from 10×10 matrices as a function of rank, for outputs corresponding to all 2^{10} inputs.** $\tilde{K}_o$ in **(a)** shows the average output complexity, while $\tilde{K}_i$ in **(b)** shows the average output complexity where each output is weighted by its frequency, i.e. the fraction of inputs that correspond to it. Figure **(c)** shows the ratio $\tilde{K}_o/\tilde{K}_i$. Red lines mark the average values in all violin plots.



(a) (b) (c)

Figure 2.17: **Output complexity of random $\{-1, 1\}$ matrix maps made from 10×10 matrices, versus matrix sparsity, for outputs corresponding to all 2^{10} inputs.** $\tilde{K}_o$ in **(a)** shows the average output complexity, while $\tilde{K}_i$ in **(b)** shows the average output complexity where each output is weighted by its frequency, i.e. the fraction of inputs that correspond to it. Figure **(c)** shows the ratio $\tilde{K}_o/\tilde{K}_i$. The darkest red lines represent the average $\tilde{K}_o$, $\tilde{K}_i$ and their ratio $\tilde{K}_o/\tilde{K}_i$, and the lighter shades of red represent one and two standard deviations.

complexity of a matrix map when at extreme values of either low rank or high sparsity. Even in these cases, the ratio $\tilde{K}_o/\tilde{K}_i$ remains around 1. This means that within all the outputs produced by these low-rank or high-sparsity matrix maps, there is no bias towards producing simpler outputs with a higher probability.

This result might seem at odds with the limited complexity conditions imposed in subsection 2.6. A random matrix map with low rank or high sparsity is definitely simpler than a full-rank zero-sparsity random matrix map. Our argument is that the limited complexity condition is a scaling relationship: a matrix map with sparsity of 99% is still $\mathcal{O}(n^2)$, since the number of entries that need to be specified is a fraction of the total n^2 entries.

On the other hand, we have shown that one can define simple matrix maps without resorting to low-rank or sparse matrices, by using circulant $\{-1, 1\}$ matrices with a small number of positive entries on each row – using approximately $\log_2 n$ elements on a $n \times n$ matrix. And as mentioned above, the matrix maps from this ensemble can be defined with $\mathcal{O}(\log n)$ bits of information. While in the previous chapters we mainly consider maps of fixed complexity, if maps grow only as $\log_2 n$ then for large enough n the requirement that $K(f) \ll K(x)$ should always hold. However, we only tried an extremely small range of $\log_2 n$ so these conclusions are very preliminary. Again, as discussed earlier, we find clear simplicity bias phenomenology even when we may not quite be in the limit of $K(f) \ll K(x)$. This suggests that the large n asymptotic behaviour persists down to smaller maps.

In the following chapters, we will test the predictions about Kolmogorov complexity and simplicity bias discussed in this chapter, as well as the structural properties of genotype-phenotype maps discussed in Chapter 1, for GP maps modelling gene regulatory networks. As a first approach, we will represent these networks in a very abstract way, similar to the matrix maps discussed above: as Boolean threshold networks.

Chapter 3

Boolean Threshold Models for Gene Networks

> "*Organic evolution has its physical analogue in the universal law that the world tends, in all its parts and particles, to pass from certain less probable to certain more probable configurations or states. This is the second law of thermodynamics. It has been called the law of evolution of the world; and we call it, after Clausius, the Principle of Entropy, which is a literal translation of Evolution in Greek.*"
>
> — D'Arcy Wentworth Thompson, *On Growth and Form*

3.1 Boolean networks as models of gene regulation

A Boolean network (BN) consists of a discrete number of Boolean variables which influence each other through an equal number of Boolean functions [114]. Typically, these Boolean functions take the state of all variables at a given time t as an input, and determine the state of the entire network at time $t+1$. As different Boolean functions will "connect" the variables in different way, the variables can be seen as nodes in a network [86].

The study of Boolean networks as simplified models of gene regulatory networks began with Stuart A. Kauffman in 1969 [114]. In Kauffman's original model, a NK network is made of N nodes, each one with K connections, that is, each one being regulated by a Boolean logic function on the states of K nodes in the network, such as:

$$S_1(t+1) \;=\; (\; S_1(t) \;\texttt{AND}\; S_2(t) \;)\; \texttt{OR NOT}\; (\; S_3(t) \;\texttt{AND}\; S_4(t) \;) \tag{3.1}$$

where $S_i(t)$ is the state of node i at time t. The state of every node in the network is then updated synchronously with all other nodes in the network. The requirement of synchronic-

ity is sometimes dropped, in what is called asynchronous updating. That said, throughout this work, we will use synchronous updating. Figure 3.1 shows an example of this kind of network, for $N = 3$ and $K = 3$, showing how different initial configurations of the network can lead to different attractors in its state space.

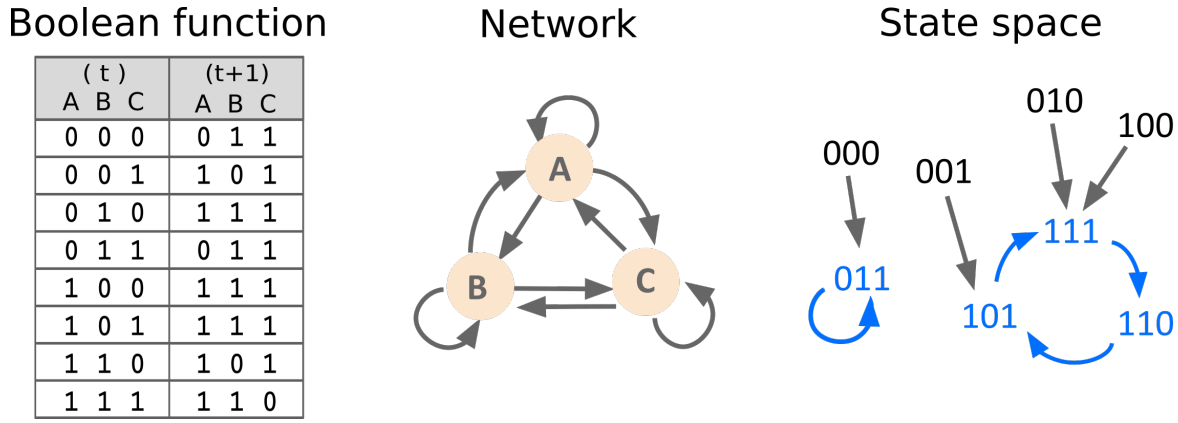
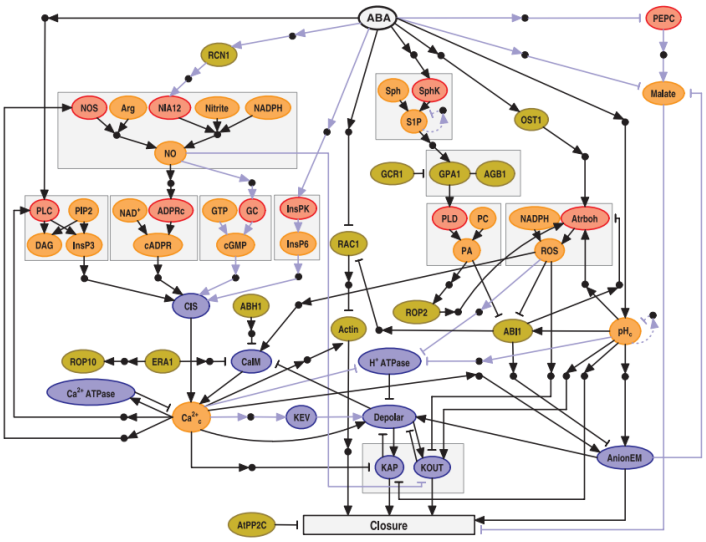


Figure 3.1: **Example of Boolean network with $N = 3$, $K = 3$.** In this network, all nodes are regulated by the whole network ($N = K$). The table on the left defines the Boolean functions that describe how the state of nodes A, B and C will change according to their states in the previous time step. The $2^3 = 8$ states in state space are organised in two attractors, one fixed point and one 3-cycle, both shown in blue.

The idea that the network might have different fates depending on its initial state is often explored in papers on cell differentiation [120], where a gene network moving towards different attractors represents a cell differentiating in distinct ways. In this case, a pluripotent cell might be represented by a gene network with many attractors, where small changes in its initial state might lead the gene network into completely different fates.

Even though they are obviously very simplified models, BNs have been widely employed to reproduce patterns of gene regulation in a variety of biological systems. Examples of gene networks studied with BNs are the GRNs regulating flower development [74], segment polarity in *Drosophila* [9], signal transduction in human fibroblasts [105], and mammalian cortical development [87]. Figure 3.2 shows an example of a BN describing a gene network responsible for plant cell signalling [149]. Boolean networks have also been used to study structural properties of gene networks, such as robustness [16, 47, 48, 214] and evolvability [69, 21, 47, 48, 214].

Boolean networks are known to be very versatile, being used to represent complex systems at multiple scales, ranging from gene regulatory networks to ecosystems [119]. This versatility, however, comes at the price of having a large parameter space. As each node in



Node	Boolean Regulatory Rule
NO	NO* = NIA12 and NOS
PLC	PLC* = ABA and $Ca2^+_c$
CaIM	CaIM* = (ROS or not ERA1 or not ABH1) and not Depolar
GPA1	GPA1* = (S1P or not GCR1) and AGB1
Atrboh	Atrboh* = pH_c and OST1 and ROP2 and not ABI1
H^+ATPase	H^+ ATPase* = not ROS and not pH_c and not $Ca2^+_c$
Malate	Malate* = PEPC and not ABA and not AnionEM
RAC1	RAC1* = not ABA and not ABI1
Actin	Actin* = $Ca2^+_c$ or not RAC1
ROS	ROS* = ABA and PA and pH_c
ABI1	ABI1* = pH_c and not PA and not ROS
KAP	KAP*= (not pH_c or not $Ca2^+_c$) and Depolar
$Ca2^+_c$	$Ca2^+_c$* = (CaIM or CIS) and not $Ca2^+_c$ ATPase
CIS	CIS* = (cGMP and cADPR) or (InsP3 and InsP6)
AnionEM	AnionEM* = (($Ca2^+_c$ or pH_c) and not ABI1) or ($Ca2^+_c$ and pH_c)
KOUT	KOUT* = (pH_c or not ROS or not NO) and Depolar
Depolar	Depolar* = KEV or AnionEM or not H^+ ATPase or not KOUT or $Ca2^+_c$
Closure	Closure* = (KOUT or KAP) and AnionEM and Actin and not Malate

Figure 3.2: **Boolean network model of signalling transduction in the stomatal guard cells in *Arabidopsis*, adapted from Li et al. [149].** Nodes are color-coded by function: enzymes are shown in red, signal transduction proteins in green, membrane transport–related nodes in blue, and secondary messengers and small molecules in orange. The Boolean rules governing the network are shown in the table to the right, where the next state of each node (marked by *) is determined by the function on the right-hand side of the equation. Nodes that have only one input were not listed. A full description of the model can be found in ref. [149].

a NK network is assigned a Boolean function on the 2^K possible states of those K nodes, there are 2^{2^K} possible functions per node, times the $N!/(N-K)!$ possible combinations of K input nodes. This brings the number of possible Boolean networks to:

$$\text{possible networks} = \left(\frac{2^{2^K} N!}{(N-K)!} \right)^N \tag{3.2}$$

which rapidly grows to prohibitively large values. For instance, for $N = 3$ and $K = 3$, as in Figure 3.1, there are over 3 billion possible BNs, whereas for a network with $N = 5$ and $K = 3$ this number grows to approximately 10^{17} [86].

In this work, we will focus on a subset of the Boolean networks, namely Boolean threshold networks (BTNs). BTNs have a much simpler interaction rule, which is defined by a threshold function similar to the ones used in artificial neural networks:

$$S_i(t+1) \;=\; \begin{cases} 1, & \text{if } \sum_{j=0}^{N} w_{ij}\, S_j(t) > 0 \\ 0, & \text{if } \sum_{j=0}^{N} w_{ij}\, S_j(t) < 0 \\ S_i(t), & \text{if } \sum_{j=0}^{N} w_{ij}\, S_j(t) = 0 \end{cases} \tag{3.3}$$

where the weights w_{ij} indicate the strength and sign of the regulation of node i by node j. Note that the value of $S_i(t+1)$ when $\sum_{j=0}^{N} w_{ij}\, S_j(t) = 0$ varies across studies, having values such as $S_i(t+1) = 1$, 0, or $S_i(t)$ itself, as we do here. This choice has been shown not to have a big impact on results [178, 58, 145]. In this work, we will be using threshold functions as defined in equation (3.3), with weight values w_{ij} taken from $\{1, -1, 0\}$, respectively indicating upregulation, downregulation and lack of interaction. This choice reflects the strong assumption that the effect of activatory and inhibitory interaction between genes is essentially additive, and that all gene-gene interactions are equally strong [246].

The threshold function in equation (3.3) inevitably reduces the range of behaviours of a network [246], but it also reduces the space of all possible networks by orders of magnitude. Instead of requiring N Boolean functions to be defined, as would be the case for a Boolean network, a BTN can be completely specified by its $N \times N$ adjacency matrix w_{ij}. Since every connection can be either -1, 0 or 1, there are $3^{N \times N}$ possible networks with N nodes, a number that grows much more slowly than the number of possible BNs presented in equation (3.2).

Each network or genotype can also be thought of as a node in a metagraph where two genotypes are neighbours if they differ in only one interaction [48]. This metagraph is shown in Figure 3.3. The whole genotype space is composed of $3^{N \times N}$ gene networks, each one connected to $2N^2$ neighbours.

BTNs have also been able to successfully model gene expression, in GRNs such as the ones regulating lymphocyte differentiation [194], signal transduction in human fibroblasts [105], and the mammalian cell cycle [76]. One of the most successful applications of BTNs is in modelling the yeast cell cycle [145, 58, 59], which led to studies predicting knockout mutant phenotypes [29, 59], as well as multiple studies providing explanations for the designability and robustness of the wild type phenotype [133, 238, 30, 41, 14, 15]. We will discuss the yeast cell cycle gene network later in this chapter, using the language of GP maps.

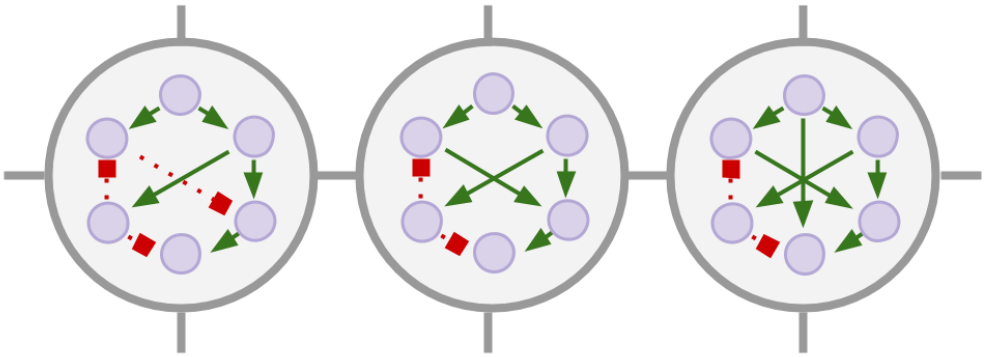


Figure 3.3: **The genotype space for GRNs.** Here, each circle contains a gene regulatory network made of six genes, with green lines marking activation and red lines indicating repression. Two gene networks, or genotypes, are neighbours if they differ by a single interaction, which is present in one network and is absent (or changes its type) in the neighbour gene network.

3.1.1 Attractors and cycles

Much of the studies of Boolean networks focus on how the connectivity of a BN affects its state space structure [113, 86]. An example of state space is shown in Figure 3.1. Each number from `000` to `111` represents a state of the network, where every digit represents the binary (Boolean) state of a node. For instance, in the state `100`, the first node is active, while the other two are inactive. Since every node can be either active or inactive, a network with N nodes has 2^N possible states in its state space. In Figure 3.1, $2^3 = 8$ possible states.

The arrows between states in Figure 3.1 indicate which state leads to which other one, according to the update rule that describes when each node will turn on or off. In the example in Figure 3.1c, state `000` leads to state `011`, which is a *fixed point*, as the update rule maps it to itself. This state space also contains a *cycle*, as state `101` leads to state `111`, which leads to state `110`, which leads back to `101`. Both this 3-cycle and the fixed point are the *attractors*, meaning that a network starting its dynamics at any point in state space will eventually end up in those regions of state space.

Biologically, attractors in BNs can have different interpretations, depending on the system that is being modelled [119]. Traditionally, when modelling gene regulatory networks, these attractors represent as alternative cell types in the organism [114, 115, 116, 118, 117], while in neural networks different attractors have been interpreted as different memories stored in the network [169, 107, 108]. Attractors can also represent different heartbeat rhythms for cardiac systems [91], or alternative species distributions when modelling ecosys-

tems [168].

3.1.2 Attractor statistics for Boolean networks

Many studies of the classical BN model have produced statistics on the number of attractors and on the distribution of basins of attraction of a network, and on how these measurements depend on parameters such as N and K [11, 244]. While most of these estimates have been obtained through simulations [11, 86], the model becomes analytically solvable for $K = 1$, when every node is regulated by a single node [82], and for $K = N$, when all nodes are influenced by the whole network. For $K = 1$, the average number of cyclic attractors has been claimed to be independent of N, with median cycle lengths of order $\sqrt{\frac{N}{2}}$ [19, 86]. At the other extreme, when $K = N$, the average number of attractors grows proportional to N, and the average cycle length grows proportional to $2^{N/2}$ [62].

The most interesting dynamics lies between both extremes: when $1 < K < N$. Statistical studies have provided increasingly accurate estimates for the average number and length of cyclic attractors [114, 19, 11, 86, 172], showing that they increase faster than any power law on N [172].

Besides the studies on attractor number and length, there is a long history of work on the stability of these attractors in state space. The stability of a network is typically measured using the time evolution of the normalised Hamming distance h_t between networks. If h_t is very small, one can approximate it by $h_{t+1} = \lambda h_t$, where λ is called the *sensitivity* of the network[1]. As a change in one node propagates on average to λ other nodes, networks with $\lambda < 1$ are said to be *stable*, or in the *ordered regime*, since a perturbation in one node is likely to vanish after a couple iterations. Conversely, networks with $\lambda > 1$ are said to be *unstable*, or in the *chaotic regime*, as any perturbation on a state might lead the network to a different attractor. For the NK model [115], the ordered and chaotic regimes happen when the average node degree K is below or above a critical value K_C. For a random NK network, where the logic function outputs are 1 (or 0, without loss of generality) with a probability p, $K_C = 1/\left[2p(1-p)\right]$. When $p = 0.5$, that is, when the logic functions are equally likely to output 1s or 0s, $K_C = 2$ [64, 63, 11]. This value is the same for scale-free Boolean networks following a degree distribution of $P(K) \propto K^{-\gamma}$ [10].

[1]The sensitivity of a network has also been defined as $S = (dh_{t+1}/dh)|_{h=0}$ for BTNs, with similar results [246].

Networks with $\lambda = 1$ (or $K = K_C$) are called *critical*, or in the *edge of chaos*, a phrase coined by the mathematician Doyne Farmer and used by Kauffman to describe systems between order and disorder [119, 120]. Mitchell Waldrop describes the edge of chaos as when "*the components of the system never quite lock into place, yet never dissolve into turbulence, either. These are the systems that are both stable enough to store information, and yet evanescent enough to transmit it. These are systems that can be organized to perform complex computations, to react to the world, to be spontaneous, adaptive, and alive.*" [236] There is evidence that many living systems evolve towards this regime, which would include GRNs [119]. There is a large body of work on criticality in BNs [115, 64, 63, 11, 10, 17, 221].

In the next sections, we will focus on Boolean threshold networks, which also display this phase transition between ordered and chaotic behaviour [129, 196, 197, 217, 109, 12]. However, a full discussion of the detailed differences between BTNs and BNs falls out of the scope of this thesis. Instead, we will study the GP map from the topology of a GRN to its dynamical (and biologically relevant) behaviour.

3.1.3 Defining phenotypes

Provided that the genotype of a GRN can be described by its topology and represented by its adjacency matrix w_{ij}, one still needs to choose how to represent its phenotype. As discussed in Chapter 1, different definitions of phenotype might be more or less appropriate depending on what is being measured. For example, if one is modelling a circadian rhythm GRN that produces oscillating behaviour for multiple initial conditions, it might be convenient to define the phenotype of that GRN as its cyclic attractor in state space. If one is interested in the possible cell fates of a pluripotent cell, it might be better to look at the whole list of attractors. Alternatively, one might define a phenotype concerning a specific set of initial conditions, or the transitions between certain states. This is the case for the fission yeast cell cycle model in ref. [58], illustrated in Figure 3.4. One could in principle then define this network's phenotype as a set of nine state transitions, namely `1001100100` $\rightarrow$ `0101100100` $\rightarrow$ `0000000100` $\rightarrow$ `0010000100` $\rightarrow$ `0010000010` $\rightarrow$ `0010001010` $\rightarrow$ `0010011010` $\rightarrow$ `0000010011` $\rightarrow$ `0001100101` $\rightarrow$ `0001100100`, which represent gene expression patterns in different phases of the fission yeast cell-division cycle. In this case, it would also be necessary to define alternative phenotypes, and a way to compute them starting from any genotype. Overall, it is hard to provide a generic model for this GP

map, since the most appropriate phenotype definition is heavily dependent on the biological problem of interest.

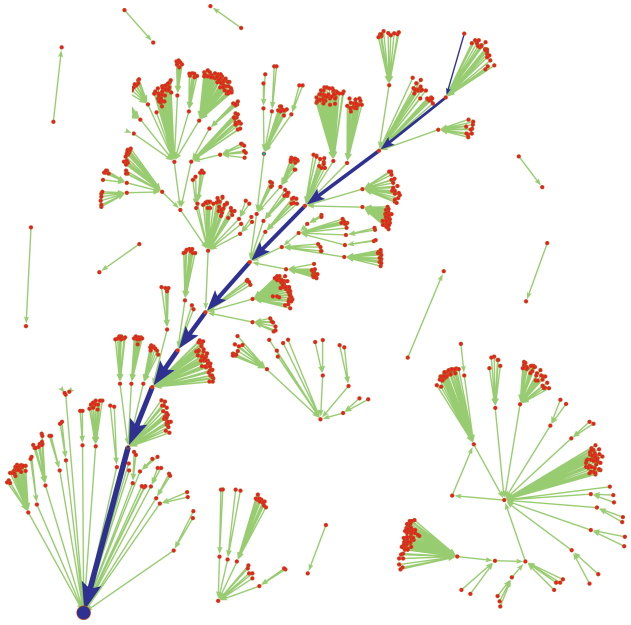


Figure 3.4: **Representation of the state space for Davidich and Bornholdt's fission yeast cell-division cycle model [58, 59].** In this figure, every circle represents one of $2^{10} = 1024$ states of the network, and every arrow represents a transition between states. Blue circles represent the states observed in the cell-division cycle of the wild-type yeast, and blue arrows represent the transitions between each those states.

Having considered the difficulty in finding one ideal definition of phenotype for this system, we instead focus on some generic phenotype definitions, that capture aspects of different kinds of biologically relevant phenotypes. In particular, we explore three different definitions of phenotype. In order, we define the phenotype of a GRN as all the attractors in its state space, then as the attractor occupying the largest fraction of its state space, and finally as a list of the attractors corresponding to all possible initial conditions of the network.

The three definitions of phenotype described above are very generic, and might not be suited for all GRNs. For instance, in GRNs where genes are named, attractors that would otherwise be equivalent under gene permutation might represent different biological behaviours. It might also be the case that not all initial conditions of the network are relevant.

In spite of these limitations, we find that all phenotype definitions produce the same generic behaviours observed for other GP maps in Chapters 1 and 2. These behaviours

include a very uneven distribution of genotypes over phenotypes, a strong bias towards low complexity phenotypes, a degree of robustness which is much larger than what would be expected for a random uncorrelated GP map, and a positive correlation between robustness and evolvability. Despite the ambiguities regarding the definition of phenotype, we also show, for the yeast cell-division cycle GRN, that the neutral network corresponding to the wild-type phenotype is larger than the average neutral network for this system. This suggests the wild-type phenotype would be more easily discovered by random mutations than most other phenotypes for this GRN.

3.2 Phenotype 1: all attractors

In this section, we define the phenotype of a GRN as all of its attractors in state space. When defining phenotypes as sets of attractors, we take into account how multiple attractors might be equivalent under symmetry operations. These operations include permuting gene order, shifting the cyclic attractors by any number of steps, or swapping 1s for 0s for a given gene. This last form of symmetry has been described as a "gauge symmetry" in work on Boolean networks by Ciliberti et al [47, 48]. Figure 3.5 shows examples of these symmetry operations. As a result, the effective number of attractors is reduced in a factor of up to $L \cdot N! \cdot 2^N$, which, for a 3-cycle attractor in a $N = 4$ network, counts for over 1000 equivalent attractors.

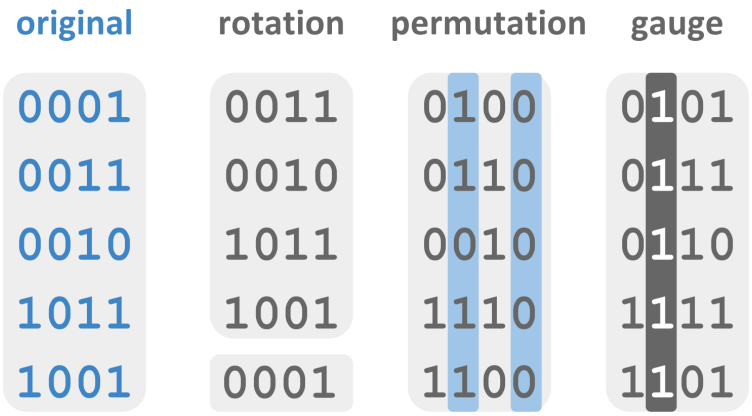


Figure 3.5: **Different symmetries of a cyclic attractor.** From left to right, the figure shows the original cycle, 0001 → 0011 → 0010 → 1011 → 1001, followed by a rotation, where the cycle is shifted by one state, a permutation, where two genes are swapped but the cycle stays in the same order, and a gauge symmetry, switching active and inactive states. We define all attractors which are equivalent up to a combination of rotations, permutations and gauge symmetries as the same attractor.

For some of the analyses in the next sections, it is also necessary to calculate the com-

plexity of an attractor. This can be done by "stacking" all the binary representations of its states into a single string, and measuring its complexity using the c_{LZ} method described in Chapter 2. For instance, the 3-cycle `1001` $\rightarrow$ `0001` $\rightarrow$ `0011`($\rightarrow$ `1001`) can be written as `100100010011`. Naturally, equivalent (up to permutation) representations of the same cycle will yield different strings, of different c_{LZ} complexity. This is addressed in Figure 3.12, where we compare the results of using different complexity measures.

It is important to notice that the symmetry operations applied to the outputs of Boolean GRNs do not introduce simplicity bias. To illustrate this point, consider a Boolean GRN GP map where all cyclic attractors of a given length L are produced with equal probability, and no other attractors are produced. For a network with N genes and 2^N states, this map would produce $2^N \times 2^{N-1} \times \cdots \times 2^{N-(L-1)}$ possible L-cycles, all with the same probability. By construction, this map is not biased towards any output, simple or complex.

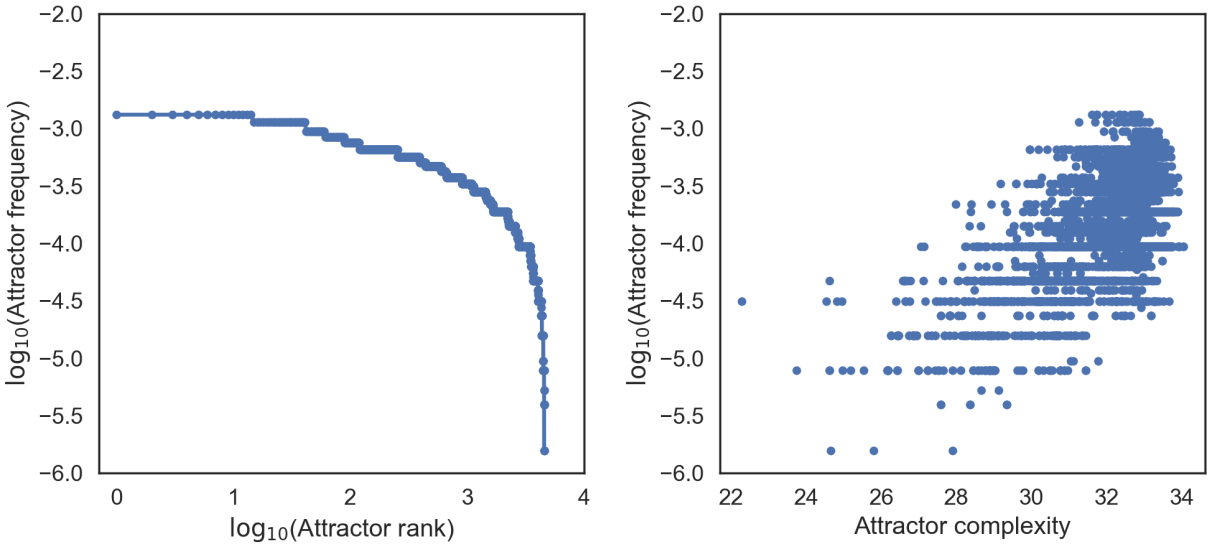


Figure 3.6: **(left)** Rank plot for the number of 6-cycles in an $N = 4$ GRN that are equivalent under symmetries. **(right)** Number of equivalent 6-cycles versus the average Kolmogorov complexity of equivalent 6-cycles. After the application of symmetry operations, this map appears to have a bias against simplicity.

Now, if one were to then apply to this map the symmetry operations described in the beginning of this section, the effect would be to "merge" all attractors equivalent under any symmetry. The result is illustrated in Figure 3.6 for $L = 6$ and a $N = 4$ GRN: once attractor symmetries are considered, more complex attractors become up to approximately 1000 times more likely than simpler attractors. This can be explained by the pigeonhole principle mentioned in section 2.2.2. In this context, the pigeonhole principle implies that

there are many more sequences of Boolean states (i.e. sequences of binary strings) that can be part of a complex attractor than there are sequences of Boolean states that can together make a single attractor.

From the example above, we can conclude that a Boolean GRN GP map that produced all possible 6-cycles with equal probability would appear to have a bias *against* simplicity, since attractors of higher complexity are likely to end up having a higher frequency after the application of symmetry operations. This suggests that, for a GRN GP map to show simplicity bias after the symmetry operations, it would have to produce low-complexity attractors with a frequency high enough to compensate this apparent increase in the frequency of complex attractors after the symmetry operations. In other words, for simplicity bias to be visible, it would have to be strong enough to "reverse the trend" displayed in Figure 3.6.

Since we are studying structural properties of GP maps, and these properties describe how phenotypes are distributed across the whole genotype space, a thorough analysis requires an exhaustive enumeration of the whole space. This stands in contrast with many previous studies on BNs and BTNs, which focus on asymptotically large networks [114, 86, 217, 109]. In this work, we analyse the phenotypes coming from a full enumeration of all $N = 4$ BTNs.

The reason for analysing networks with as few as $N = 4$ genes is combinatorial. Even at the simplest approximation, where one considers interaction as either positive $(+1)$, negative (-1) or absent (0), the number of Boolean threshold networks grows as $3^{N\times N}$, which means that there are $3^9 = 19683$ networks with $N = 3$, $3^{16} \approx 4.3 \times 10^7$ networks with $N = 4$, approximately 10^{12} networks with $N = 5$ and 10^{17} networks with $N = 6$, at which point it becomes intractable to enumerate all gene networks.

We therefore chose to mainly study networks with $N = 4$ genes, which allowed us to take more thorough statistics of these networks, without being limited to finite samples. However, to check our results, we also provide statistics for a sample of 10^7 networks with $N = 10$, for which there are $3^{100} \approx 10^{48}$ genotypes.

Figure 3.7 shows the distribution of neutral network sizes presented by this GP map, for $N = 4$ networks. There are 8771 phenotypes, resulting in an average phenotype frequency of $1/8771 \approx 10^{-4}$. The plot, however, shows a very skewed distribution. Nearly 80% of all neutral networks are smaller than a millionth of genotype space, while the most designable phenotype covers 26% of genotype space.

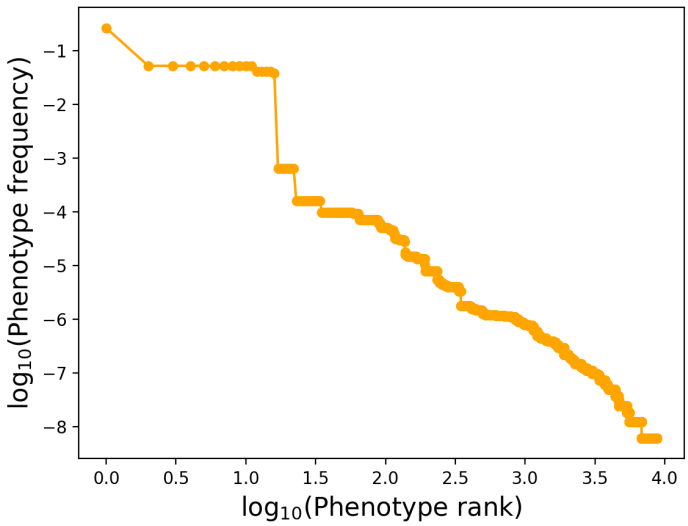


Figure 3.7: Rank plot for the number of gene networks that produce each one of all the 8771 phenotypes found in the whole space of $N = 4$ gene networks. It is a very uneven distribution, with 99% of all genotypes mapping to 0.38% of all phenotypes, and the most designable phenotype corresponding to 26% of the genotype space.

In addition to the rank plot, we used both the $N = 4$ and $N = 10$ data to study the distribution of the number of attractors per network, of attractor length, attractor complexity and basin size. Both the number of attractors per network and the attractor length are distributed in a roughly exponential fashion. The probability of a BTN with n of attractors decays as $e^{-1.24n}$ for $N = 4$ as $e^{-0.51n}$ for $N = 10$, and the probability of a cyclic attractor with L states falls as $e^{-1.40L}$ for $N = 4$ as $e^{-0.32L}$ for $N = 10$. This is shown Figure 3.8a and 3.8b.

The $N = 10$ networks dataset also shows an interesting behaviour as networks with 32 and 64 attractors are much more likely than the extrapolation from the exponential decay would suggest. This can be explained by the symmetry of the network state space: every state of a N-node Boolean network can be seen as a vertex of an N-dimensional cube, which can be "sliced" in different ways, producing hyperplanes with 2^j states, where j is between 0 and N. In the network, this "slicing" happens when a node-node interaction only affects the behaviour of a set of j genes, thus dividing the state space into 2^j-sized fractions[2].

While we are considering this BTN as a GP map, it can also be viewed simply as an

[2]For example, a $N = 10$ network with a single negative interaction from node 1 to itself would make all states in which the first gene is active, i.e. all '1*' states, lead to almost identical states where the first gene is inactive, that is, all '0*' states. These '0*' states will then be fixed points. In this way, this network would produce $2^{(10-1)} = 512$ attractors. Following this logic, a network with 4 independent self-repressive interactions would produce a phenotype with $2^{10-4} = 64$ attractors.

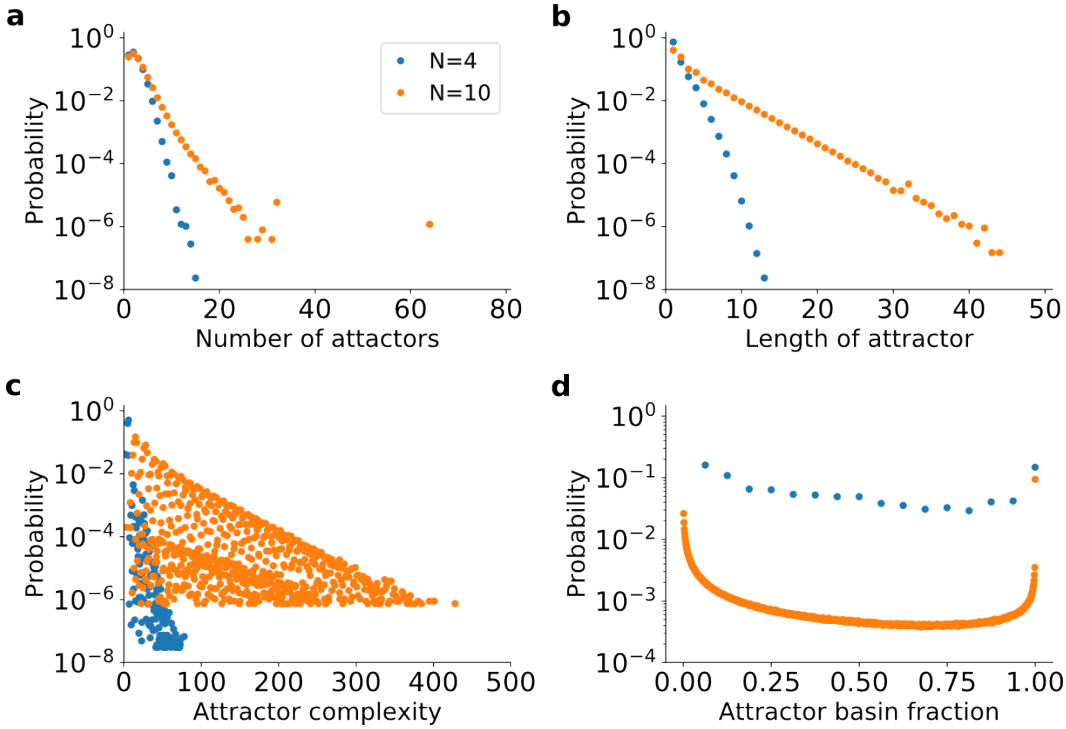


Figure 3.8: **Statistics for the attractors of $N = 4$ and $N = 10$, shown respectively in blue and orange. (a)** The probability that a BTN produces n attractors decays as $e^{-1.24n}$ for $N = 4$ as $e^{-0.51n}$ for $N = 10$. Note the higher probability for $n = 32$ and $n = 64$ for the $N = 10$ data, which is explained in the text. **(b)** The probability of a cyclic attractor with L states decays as $e^{-1.40L}$ for $N = 4$ as $e^{-0.32L}$ for $N = 10$. **(c)** Both systems exhibit simplicity bias, as the upper bound for attractor probability decreases exponentially with attractor c_{LZ} complexity. **(d)** The fraction of state space taken by different attractors is distributed non-monotonically.

input-output map, as studied in Chapter 2. In this case, it is a map from input parameters (network topologies) to outputs (attractors in state space). Since the procedure of producing attractors from a network does not become more complex with n (it might take longer to produce all outputs, but the procedure is the same), this map satisfies the condition of limited complexity described in Chapter 2, meaning that $K(f) + K(n) \ll K(x) + \mathcal{O}(1)$, where $K(f)$ stands for the complexity of the map, n is the input length, and x is any output of the map. With that in mind, one would expect this input-output map to show simplicity bias, as predicted in Chapter 2. Figure 3.8c shows this map behaves as expected: the probability of an attractor is bounded by its c_{LZ} complexity, with the exponential upper bound following $P(x) \leq 2^{-a\tilde{K}(x)-b}$ with $a = 0.319$ and $b = 1.946$ for $N = 4$ and $a = 0.043$ and $b = 2.844$ for $N = 10$.

Figures 3.8a and 3.8b already imply something like simplicity bias, since they show that complex attractor landscapes with many attractors or else long attractors are both

exponentially less likely to occur. The exponential scaling is quite tight, suggesting that there may be simple combinatorial arguments that could explain them. In contrast, the simplicity bias shown in Figure 3.8c, shows a less tight scaling, but it is much simpler to derive, requiring only the arguments presented in Chapter 2.

Figure 3.8d shows the distribution of the fraction of state space taken by the basin of attraction of each attractor. Both $N = 4$ and $N = 10$ networks show attractors with all sizes from 1 to 2^N, but in both cases there is a peak both at the minimum and the maximum fractions of the state space. That is, the majority of attractors either correspond to most initial states, or to only a few. Overall, this implies that a network placed in any initial condition is likely to either remain there, or to fall into an attractor that takes over most of state space.

3.3 Phenotype 2: dominant attractor

In this section, we define the phenotype of a GRN as its dominant attractor, that is, the attractor taking the largest fraction of its state space. We present the biases found in this genotype-phenotype map, for the same full enumeration of $3^{16} \approx 43$ million $N = 4$ networks, as well as for a sample of 10^7 networks with $N = 10$. Echoing what has been seen for other GP maps, we find that the distribution of neutral network sizes is very biased, with most phenotypes corresponding to less than 1% of all genotypes, and a few low-complexity phenotypes covering most of genotype space. Finally, we compare the phenotype neutral networks in this GP map to the ones presented in ref. [99], by looking at their robustness to mutations.

Identifying the dominant attractor

To identify the attractor with the largest basin for each genotype, we use the same method as Nochomovitz and Li [178]: first, we use the update function to produce a list of 2^N "next states", which we then connect as a directed graph from state to state, as illustrated in Figure 3.1. We then use Tarjan's algorithm to calculate the *strongly connected components* of this directed graph, that is, the sections of state space where every state can reach every other one. Since the threshold update function is deterministic, every state will only lead to one other state, meaning that strongly connected components of the state graph can only be the attractors of the BTN.

The distribution of neutral network sizes is not uniform

The distribution of neutral network sizes (NNS) in this map is very biased. Figure 3.9 shows the NNS for all 2759 phenotypes observed for $N = 4$ networks, and although there are over 43 million genotypes, 69% of the attractors are only found in less than a millionth of the space, while the most common attractor, which is a single fixed point, is found in 67% of genotype space.

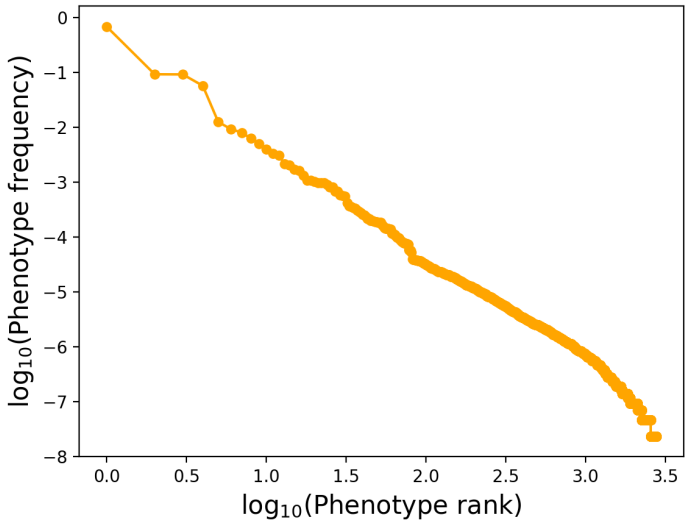


Figure 3.9: Rank plot for the number of gene networks that produce each one of all the 2759 dominant attractors found in the whole space of $N = 4$ gene networks. It is a very uneven distribution, with 99% of all genotypes mapping to 0.38% of all phenotypes, and the most designable phenotype corresponding to 26% of the genotype space.

This uneven distribution of neutral network sizes is also observed within cyclic outputs of the same length, as already noted by Nochomovitz and Li for Boolean threshold networks of the same size [178]. Figure 3.10 shows this pattern for cyclic attractors of lengths $L = 4$ to 7 in $N = 4$ networks. For all attractor lengths, the distribution spans over orders of magnitude, with its mean decreasing exponentially as L grows.

Simplicity bias

The negative relation between mean neutral set size and cycle length L is part of a wider pattern: this GP map shows simplicity bias. Figure 3.11 shows some evidence of this bias, showing the six most common phenotypes and six of the least common phenotypes for a network of $N = 10$ genes, all followed by the number of genotypes that map to each phenotype. In the figure, each square represents a cyclic attractor, where every column

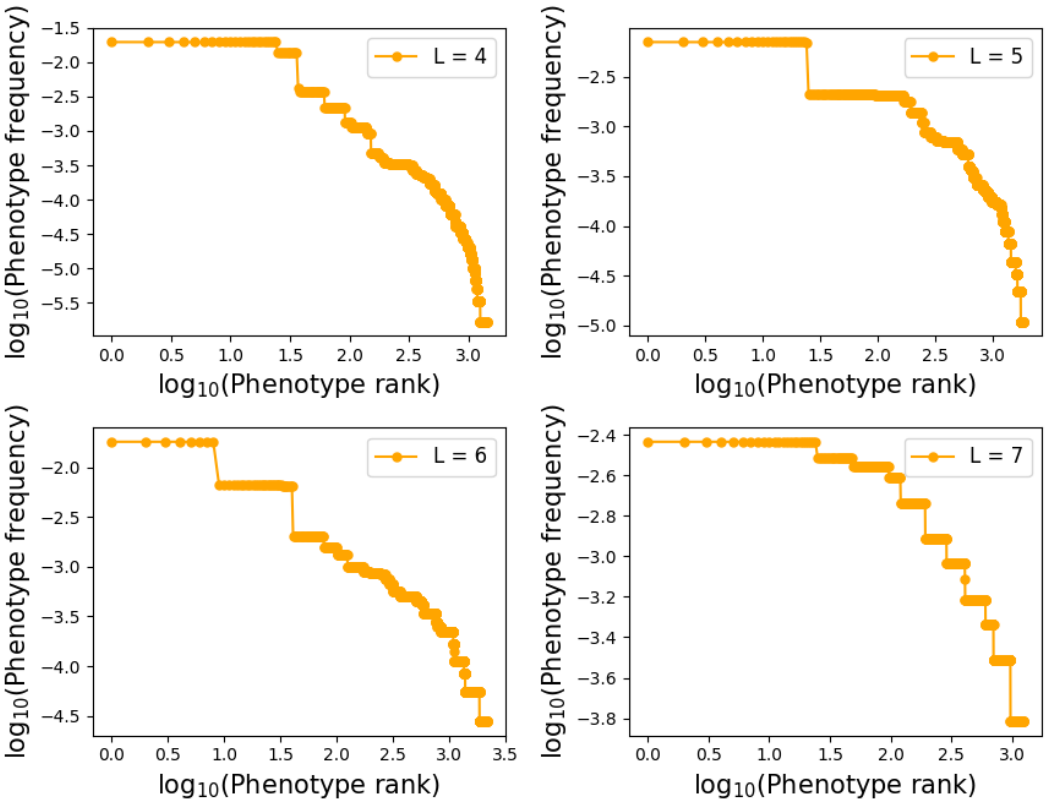


Figure 3.10: The skewed distribution of neutral network sizes is also observed within cyclic outputs of the same length, shown here for lengths $L = 4$ to 7, for $N = 4$ networks.

represents the binary state of a node, with yellow for `1` and blue for `0`; for example, the top left square indicates the 2-state cycle `1000110001` → `1110001110`, produced by 6751 genotypes, while the bottom right network indicates a 16-state cycle produced by a single network. The most common phenotypes, shown on top, are visibly simpler: apart from being shorter cycles, they involve fewer gene activations/inactivations than the least common phenotypes, shown in the bottom.

While it is clear from Figure 3.11 that the most common attractors are shorter, and subjectively simpler to the naked eye, that is obviously no replacement for actually measuring the simplicity bias in this GP map. Nochomovitz and Li measure the length of a cyclic attractor as a proxy for its complexity [178], but this measure does not address the uneven distribution that also happens among the cycles of the same equal length, as shown in Figure 3.10. Since two cycles with the same length can represent two different patterns of gene activation, a complexity measure should encompass these differences. We quantify these differences in complexity between different attractors by "stacking" all states of the cycle in a single binary string, as described in section 3.1.3, and estimate the Kolmogorov

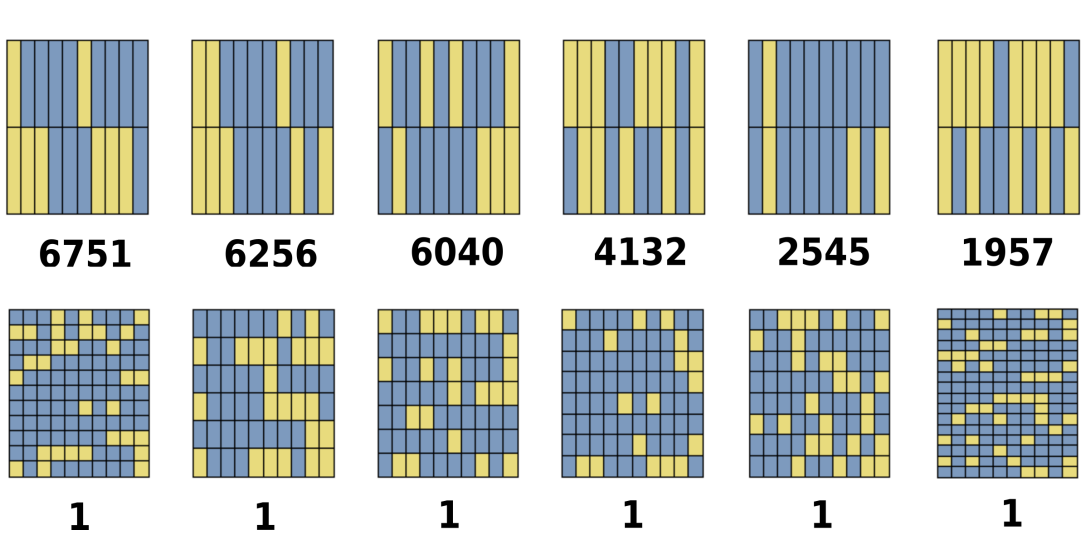


Figure 3.11: **Most and least common cyclic dominant attractors for a network of $N = 10$ genes.** Each square represents a cyclic attractor, where every column represents the binary state of a node, with green for `1` and grey for `0`. For example, the top left square indicates the 2-state cycle `1000010000` $\rightarrow$ `1110001110`, produced by 6751 genotypes, while the bottom right network indicates a 16-state cycle. The numbers shown below each attractor indicate its neutral set size.

complexity of the resulting string using the canonical Lempel-Ziv method described in Chapter 2. In addition to the attractor length L and to the Lempel-Ziv c_{LZ} complexity, we also measured the total Hamming distance accumulated through the cycle: as one state evolves into another, a number of genes has to be turned on and off, that number corresponding to the Hamming distance between both states. This distance, accumulated over all steps in a cyclic attractor, can also be used as an estimate of the complexity of that attractor.

These three proxies for phenotype complexity are compared in Figure 3.12. The three top subplots show cycle length, total Hamming distance and c_{LZ} plotted against each other for multiple attractors, showing clear correlation between all measures. The bottom plots show the three proxies plotted against the neutral network size of every phenotype, all showing the same behaviour: the number of genotypes mapping to a phenotype is bounded by an exponentially decreasing function on complexity – a hallmark of simplicity bias, as discussed in Chapter 2.

Figure 3.12 also shows an important difference between this definition of phenotype, where only the dominant attractor is taken into account, and the previous definition, where all the attractors produced by a GRN counted as phenotypes. As shown in the bottom left plot of Figure 3.12, the length of the dominant attractor does not follow a power law

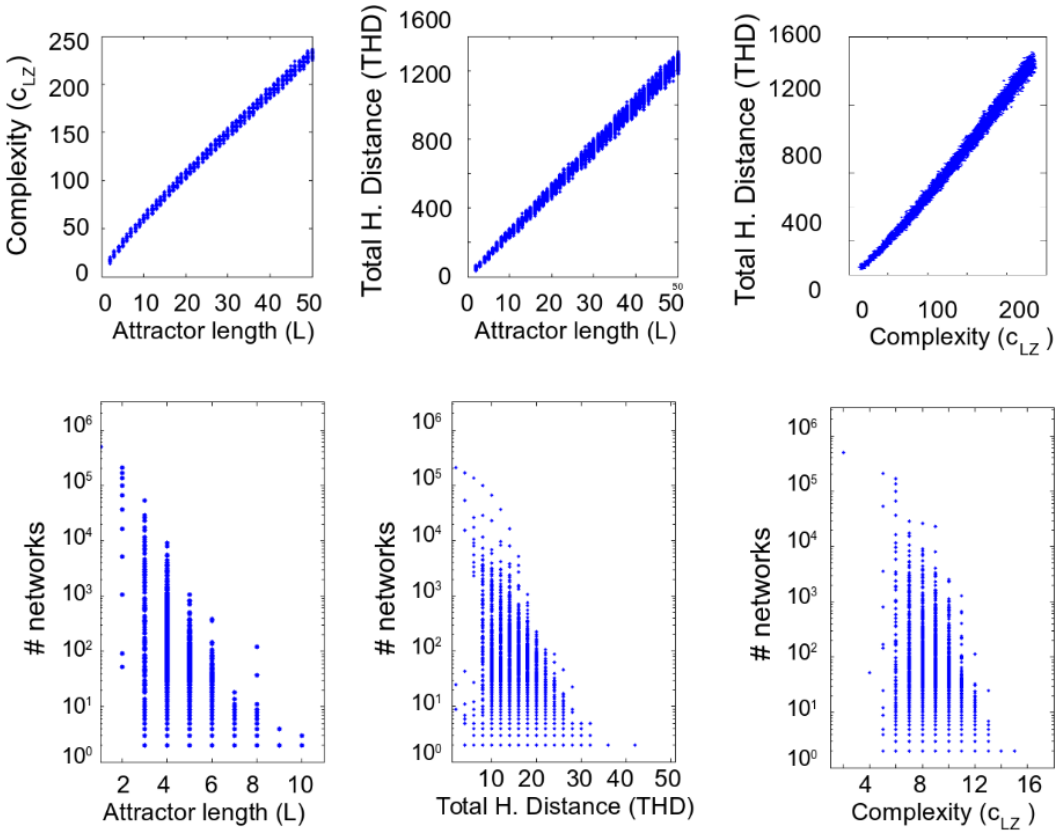


Figure 3.12: **Simplicity bias according to multiple estimates of attractor complexity.** The three top plots show attractor length, total Hamming distance and Kolmogorov complexity plotted against each other. The bottom plots show these three proxies for complexity plotted against the neutral network size of every phenotype, which is bounded by an exponential decay for the three measures.

distribution, as it did for phenotype 1. Instead, it is bounded by an exponential distribution, similar to the other estimates of complexity presented in the figure.

Given that the correlation between different measures of complexity is so evident, it could be the case that the simplicity bias measured with c_{LZ} is simply revealing the bias towards shorter attractors, and nothing else, which would make it redundant to use both measures. Figure 3.13 indicates that is not the case. It is a heat map, produced for all $N = 4$ networks, where every column represents attractors of the same cycle length, such as the ones grouped in the same the subplots in Figure 3.10. In the figure, the darker shades of red are always in the lower, low-c_{LZ} side of the red-shaded part of every column. Since darker colours represent more genotypes, this plot indicates that more genotypes map to phenotypes of lower complexity, even when comparing phenotypes of the same attractor length.

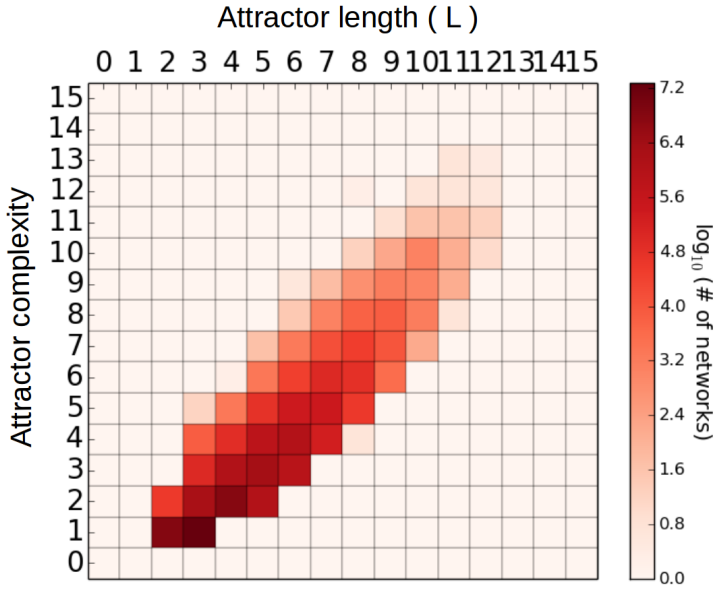


Figure 3.13: Heat map for $N = 4$ networks, comparing the Kolmogorov complexity of attractors to their cycle length. As darker colours represent more genotypes, this plot indicates that more gene networks produce phenotypes of lower complexity, even when comparing phenotypes of the same cycle length.

In addition to having a highly biased distribution of neutral network sizes and a strong bias towards simple outputs, this GP map also shows evidence of a very correlated neutral network landscape. As shown in Figure 3.14 for all $N = 4$ networks, robustness scales with the logarithm of phenotype frequency, similar to the self-assembly GP maps presented in ref. [99], also shown in Figure 3.14.

We also confirm that this map shows shape space covering, meaning that most phenotypes are found within a couple mutations of any genotype. While any $N = 4$ GRN is at most 16 mutations away from other genotypes, corresponding to the number of entries in the adjacency matrix w_{ij}, the minimum number of mutations required to transform any phenotype into any other is notably smaller: for $N = 4$, the diameter or maximum path length is of 3 mutational steps between phenotypes. The average path length, taken over samples of 10^4 pairs of phenotypes, is of 1.98 ± 0.15 steps, which is also far from the maximum $N^2 = 16$.

It is important to question the assumptions that were made in building this GP map model. For example, in defining the phenotype of a network as its largest attractor, we lose information about which initial conditions lead to that phenotype, and also ignore other attractors present in the network state space. For a gene network where all biologically relevant initial conditions lead to the same attractors, such as the yeast cell cycle studied

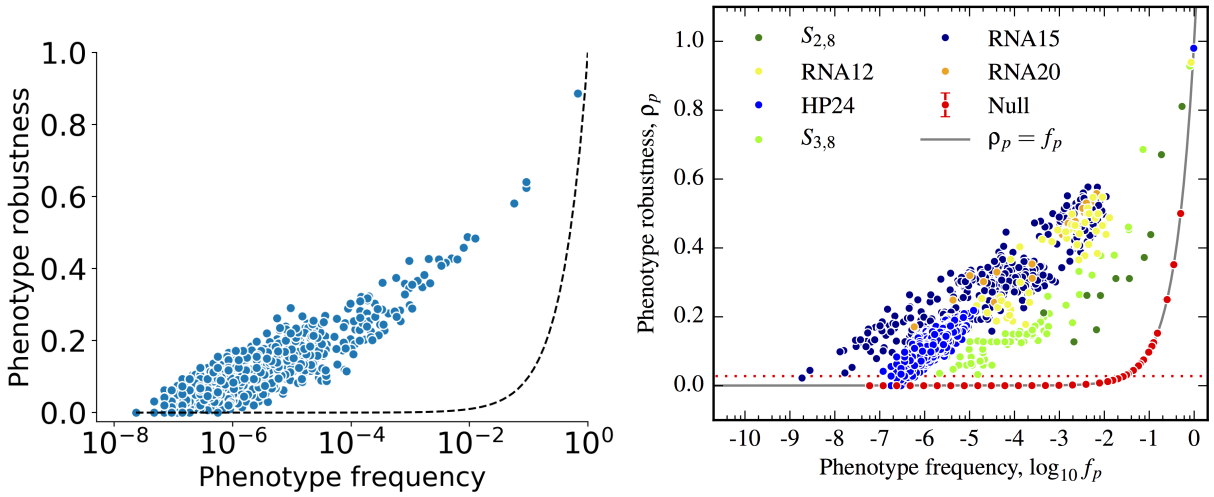


Figure 3.14: Phenotype frequency versus phenotype robustness for $N = 4$ gene regulatory networks, for the phenotype defined as the dominant attractor **(left)** and for the other GP maps studied by Greenbury et al. [99] **(right)**.

by Li et al. [145], these limitations represent no problem, but this definition of phenotype cannot represent a bistable system, or a cell that, provided the right initial conditions, might differentiate into other cell types [119].

In the next section, we provide a more fine-grained model of this GP map, and define the phenotype taking into account which initial conditions fall into which attractors in state space. Or, in biological terms, which gene expression patterns lead to which cellular behaviours.

3.4 Phenotype 3: list of attractors

In this section, we use a definition of phenotype inspired by Lomnitz and Savageau's analysis of the phenotypic repertoire of synthetic gene oscillators [155]. We define the phenotype of a N-nodes gene regulatory network as a 2^N-long string of the attractors corresponding to each one of its 2^N possible states. For example, for $N = 4$, a phenotype would be a string such as `AABACBDBCCCDDCCD`, indicating that that states `0000` and `0001` map to attractor A, state `0010` maps to attractor B, and so on.

Having the phenotype written down as a string also allows us to calculate its Kolmogorov complexity using the c_{LZ} method. We do this by running over all genotypes, and producing their corresponding strings of attractors, and translating those strings into the shortest

binary strings that can accommodate the range of attractors presented by this GP map. Since the whole $N = 4$ genotype space produces a total of 8172 different attractors, the index corresponding to each attractor can be represented in $\lceil \log_2(8172) \rceil = 13$ bits, and the whole phenotype string made of $2^4 = 16$ attractors can be represented in $16 \times 13 = 208$ bits. We can then measure phenotype complexity by applying c_{LZ} to these 208-bit strings.

As this GP map uses a finer-grained definition of phenotype, it produces more phenotypes than the more coarse-grained GP map presented in the previous section. Figure 3.15a shows a rank plot of the normalised neutral network sizes of the 3.8×10^6 phenotypes.

In addition to the bias shown in Figure 3.15a, this map also shows a clear simplicity bias, as indicated for $N = 4$ networks in Figure 3.15b. The plot exhibits the characteristic triangular shape observed for the maps discussed in Chapter 2, with the most frequent phenotype being when all initial conditions lead to `0000` as a fixed point, representing when all genes turn off and there is no more gene expression.

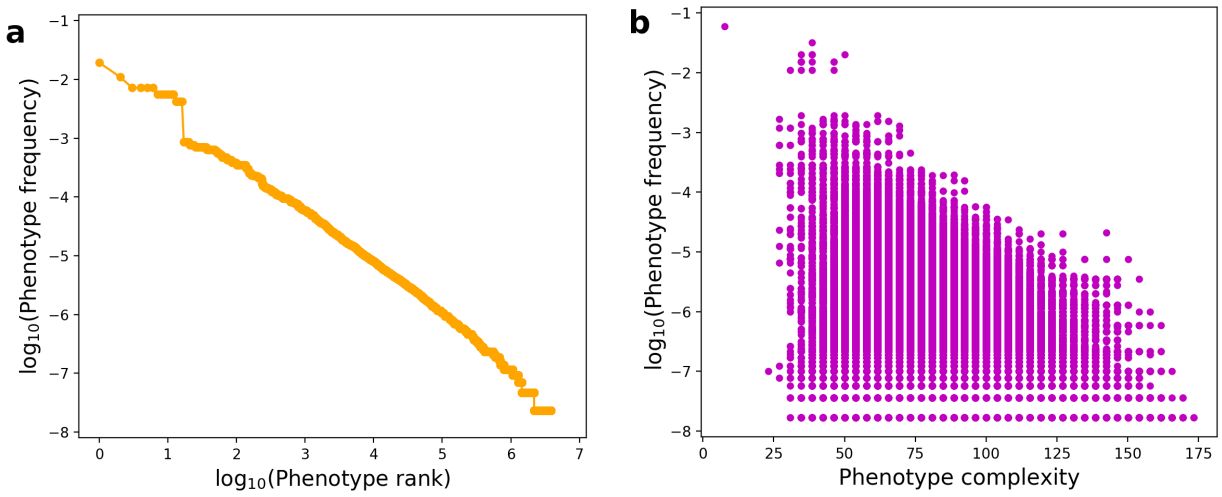


Figure 3.15: **(a)** Rank plot showing a very skewed distribution of phenotype frequency, for the phenotype defined as the list of attractors, for $N = 4$ GRNs. **(b)** Scatter plot of phenotype frequency versus complexity for the same phenotypes, showing evidence of simplicity bias. Complexity was measured using the Lempel-Ziv algorithm discussed in Chapter 2.

The other structural properties of GP maps presented in Chapter 1 are also present in this GP map. Figure 3.16a and b, plotted for all $N = 4$ networks, respectively show a negative correlation between genotype robustness and evolvability, and a positive correlation between phenotype robustness and evolvability. This map also presents a similar relation

between phenotype frequency and robustness discussed in Chapter 1: phenotype robustness (ρ_P) the logarithm of its frequency in genotype space ($\log_{10} f_p$) scale in an approximately linear fashion, with $\rho_P \approx 0.1 \log_{10} f_p + 0.75$. This is shown in Figure 3.17.

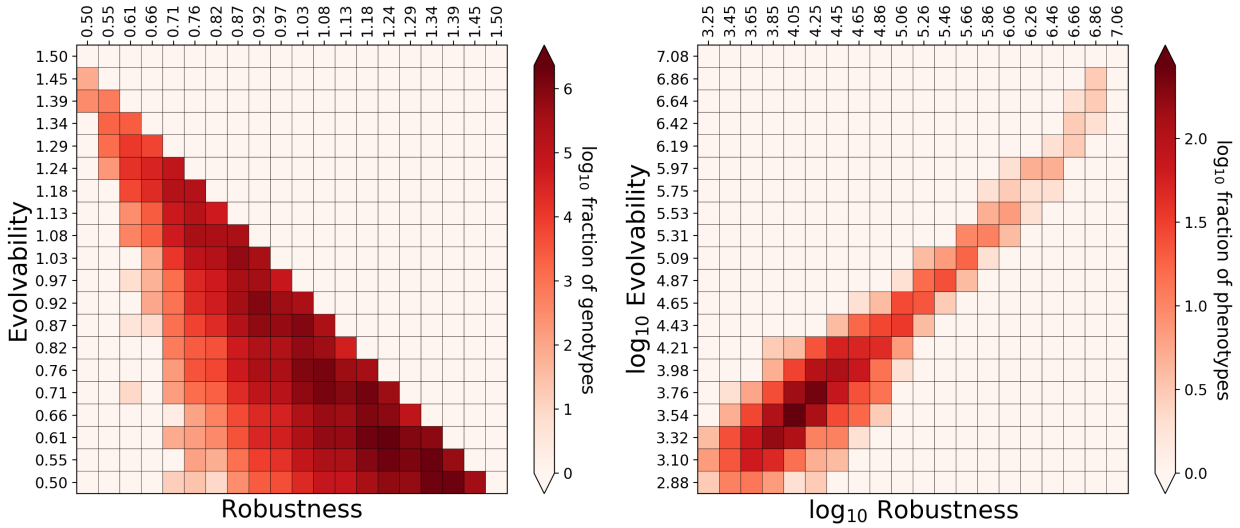


Figure 3.16: Heat maps comparing robustness and evolvability at genotype level, where they show a negative correlation **(left)**, and at phenotype level, for the phenotype defined as the list of attractors for $N = 4$ GRNs, where they show a positive correlation **(right)**.

Finally, this definition of phenotype also shows shape space covering. Again, while all $N = 4$ genotypes are at most 16 mutations away from each other, it takes much fewer mutations to transform any phenotype into any other. For $N = 4$, since there are 3.8×10^6 phenotypes and calculating the diameter of the network would become too computationally expensive, we instead calculated its pseudo-diameter, which is an approximate measure of the network's diameter or maximum path length [187]. The pseudo-diameter of the network of $N = 4$ phenotypes, for this phenotype definition is of 6 mutational steps, and the average path length, averaged over samples of 10^4 pairs of phenotypes, is of 3.8 ± 0.7 mutations. Both values are much smaller than $N^2 = 16$. The pseudo-diameter was calculated using Graph-tool, a high-performance Python library for manipulation and statistical analysis of networks [186].

3.4.1 Applications of the Neutral Network Size Estimator

As we discussed previously, the size of genotype space in this system grows as $G(N) = 3^{N \times N}$, where N is the number of genes in the GRN. While full enumeration of genotype space can be used for $N = 4$, GRNs of biological interest lie in larger genotype spaces. For instance,

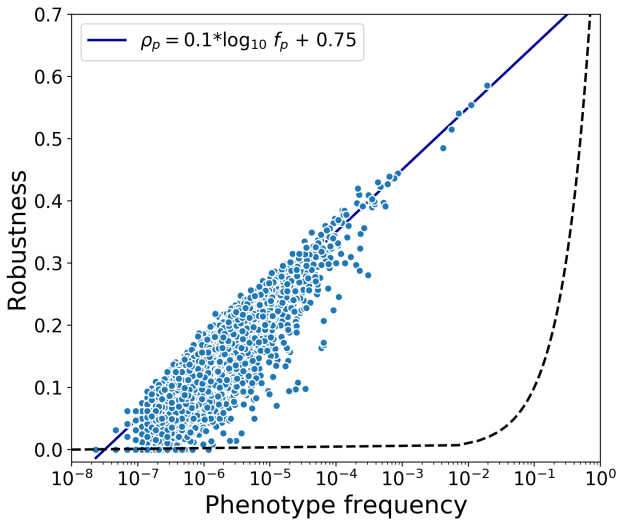


Figure 3.17: Phenotype robustness grows proportionally to the logarithm of phenotype frequency, for the phenotype defined as the list of attractors for $N = 4$ gene networks. The dashed black line represents the null model expectation $\rho_p = f_p$.

the yeast cell cycle networks explored by Davidich and Bornholdt [58, 59] and by Li et al. [145] are modelled as Boolean threshold networks made of 10 genes, which are in a space of $3^{100} \approx 5 \times 10^{47}$ possible networks. For these larger genotype spaces, even brute force sampling will be inefficient at accessing a significant fraction of the whole space. It is necessary to use other approaches to estimate phenotype neutral network sizes.

To deal with challenge of genotype spaces which are too large even for brute force sampling, we adapted Jörg et al's Neutral Network Size Estimator (NNSE) [112] for Boolean threshold networks, for the third definition of phenotype discussed in this chapter, i.e. the list of attractors. The NNSE algorithm falls within a class of Monte Carlo methods used in statistical physics to investigate complex energy landscapes for which direct enumeration techniques are too inefficient [166, 165]. The NNSE algorithm was originally developed to study the RNA map, presented in Chapter 1, but Jörg et al. also note that their approach should work "*not only for RNA genotypes, but for any genotype space (discrete or continuous) as long as a distance metric between phenotypes exists.*" As this is the case for the BTN maps studied in this chapter, the adaptation was straightforward, as described below.

The NNSE algorithm follows a *nested Monte Carlo* approach: given a discrete genotype space, and the goal of estimating the number of genotypes that produce a given phenotype x, one defines $V(r)$ as the number of genotypes which map to phenotypes x' which are within a distance $d(x, x') \leq r$ of phenotype x. The distance $d(x, x')$ is defined in phenotype space,

ranging from $d = 0$ for identical phenotypes to $d = d_{max}$ for maximally different phenotypes. In this way, $V(r)$ can be seen as the "volume" inside a shell in phenotype space, centered in x and with radius equal to r. Here we will be using $V(r)$ to refer not only to the volume of this shell, but to the shell itself, and to the set of genotypes mapping to phenotypes in the shell. These shells of radius r are concentric – hence "nested Monte Carlo".

The smallest shell is $V(0)$, corresponding to the neutral network that maps to phenotype x. Its volume can be expressed as:

$$V(0) = \frac{V(0)}{V(1)} \cdot \frac{V(1)}{V(2)} \cdot \ldots \cdot \frac{V(d_{max}-2)}{V(d_{max}-1)} \cdot \frac{V(d_{max}-1)}{V(d_{max})} \cdot V(d_{max}) \tag{3.4}$$

$V(d_{max})$ is already known: it is the size of the whole genotype space, $3^{N \times N}$. Therefore, the task of calculating $V(0)$ is split into a series of smaller and easier tasks of calculating $\frac{V(r)}{V(r+1)}$. This is done in the following manner: we perform a random walk in genotype space, starting at a random genotype G_1 in $V(r+1)$ and adding one mutation to G_t at every step t. For Boolean networks, this mutation means changing one of the elements in its adjacency matrix w_{ij}, thus producing a new genotype G'. If G' is still within $V(r+1)$, G_{t+1} is set to G', otherwise G_{t+1} is set to G_t. If this random walk is long enough, it will sample $V(r+1)$ uniformly. Given that a fraction of $V(r+1)$ is also in $V(r)$, the ratio $\frac{V(r)}{V(r+1)}$ can then be estimated from this sample.

Since this algorithm relies on random walks, it cannot access different components of the neutral network, and will provide an incorrect estimate of neutral network size for neutral networks with more than one component. We address this limitation by starting multiple random walks at different points of genotype space, and swapping genotypes in $V(r)$ and $V(r+1)$ when they both lie in the other set as well. This makes sure that the sampling is ergodic, and reaches all areas of sequence space, allowing for a more accurate estimate of the ratio $\frac{V(r)}{V(r+1)}$. These ratios can then be multiplied, providing an estimate of $V(0)$, the neutral network size of phenotype x.

The NNSE algorithm highlights parallels between the statistics of GP maps and problems in computational physics. The uniform sampling of $V(r)$ using random walks in genotype space is equivalent to the step of thermalisation in Monte Carlo simulations [237], and the step where genotypes in different shells are swapped is known as parallel tempering or replica exchange, a powerful tool for simulations of systems with complex potential energy landscapes [216, 71].

The NNSE method can be applied to any GP map, or to any definition of phenotype, provided that the following three functions are specified:

- `Mutate(G)` - produces G' from G. Here, this function picks one random entry of the adjacency matrix w_{ij} and changes it to another value within $\{-1, 0, 1\}$
- `Distance(x, x')` - Calculates the distance $d(x, x')$ between phenotypes. For this GP map, this is the Hamming distance.
- `fold(G)` - Produces phenotype from genotype. described in section 3.4.

As the original implementation of the NNSE takes an RNA structure as input, it also uses a function `inversefold(x)`, which uses the ViennaRNA package [106] to find a RNA genotype that is compatible with that phenotype. In our case, we skip that step, providing our adaptation of the algorithm with an input pair of genotype and phenotype. In the next section, we present results of applying algorithm to the GRN GP map using the list of attractors as its phenotype.

The NNSE algorithm, apart from the input phenotype (and genotype, in our case), requires a set of input parameters: `NTHERMAL`, the number of thermalization Monte Carlo sweeps to be applied before the nested Monte Carlo runs, `NRUNS`, the total number of measurements of the ratios $\frac{V(r)}{V(r+1)}$, and a seed for the random number generator. In Appendix B, we show a series of tests confirming the robustness of our adapted NNSE, showing results for different combinations of `NTHERMAL` and `NRUNS`.

3.4.2 Results using the NNSE

Before applying the NNSE to biologically relevant gene networks such as the $N = 10$ yeast cell cycle networks, it is important to confirm the algorithm works for this GP map. In this section, we compare the NNSE results to the neutral network sizes obtained by fully enumerating the space of $N = 4$ GRNs, and by partially sampling $N = 5$ and $N = 6$ networks.

Figure 3.18 shows the 3.8×10^6 phenotypes produced by all $N = 4$ networks, ranked in order of neutral network size. The NNS obtained from the full enumeration, shown in purple, is compared to the results from an NNSE estimation ran over `NRUNS`$= 10^4$ iterations,

shown in grey. Apart from a few phenotypes with small neutral networks, the match between estimation and measurement suggests the algorithm works remarkably well.

The accuracy of the NNSE for $N = 4$ GRNs suggests this algorithm might perform well on larger genotype spaces. To produce a "ground truth" for the NNSE, we took a sample of 10^6 genotypes for $N = 5$ and $N = 6$ GRNs, from a total of $3^{25} \approx 8.5 \times 10^{11}$ genotypes and $3^{36} = 1.5 \times 10^{17}$ genotypes respectively, as shown in Figure 3.19. Naturally, this sampling method is likely to miss many of the smaller neutral networks, or to overestimate their neutral network size. Its error should also be within one over the square root of the actual neutral network size, which is the same error resulting from the NNSE.

Figure 3.19 shows NNSE results for $N = 5$ and $N = 6$ GRNs, for runs of 20 trials of 5×10^3 thermalisation steps, followed by 2×10^4 iterations. In both cases, there is a strong match between the NNSE and the estimate provided by randomly sampling genotypes.

The ability to explore larger genotype spaces allows us to tackle a very interesting question: how large are the neutral network sizes of the gene networks corresponding to wild-type phenotypes? In other words, are the phenotypes like the yeast cell cycles described by Davidich and Bornholdt [58, 59] and Li et al. [145] likely to be found by random mutations?

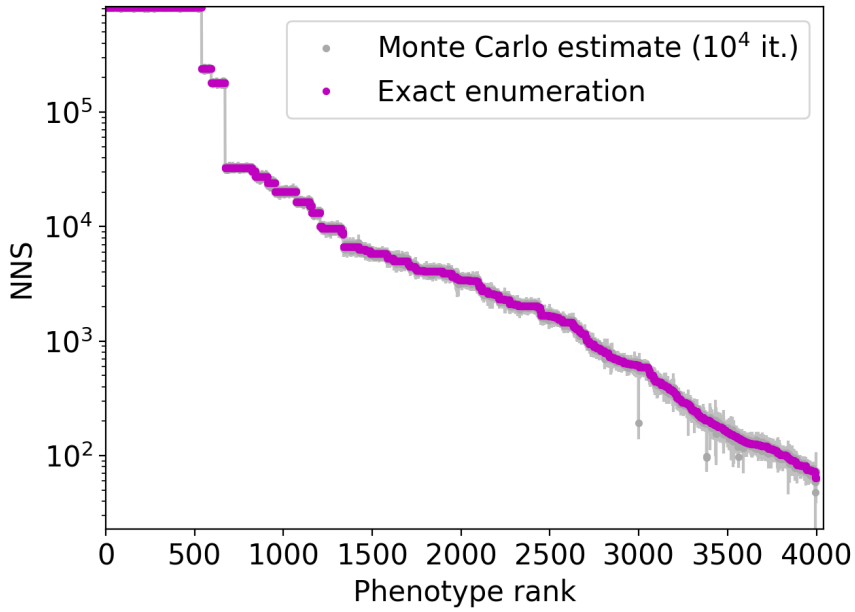
We addressed this question by estimating the neutral network size of 200 GRNs with $N = 10$ genes, plus the fission yeast cell cycle network in ref. [58]. The NNSE algorithm was run over 500 thermalisation steps, followed by 2000 runs of the nested Monte Carlo step. Figure 3.20 shows the result, indicating that the neutral network mapping to the fission yeast phenotype, indicated by the red arrow, is larger than approximately 90% of all other neutral networks, which is a rough estimate obtained from the small sample of 200 networks. The robustness of the NNSE algorithm to its parameters is discussed in Appendix B.

3.5 Discussion

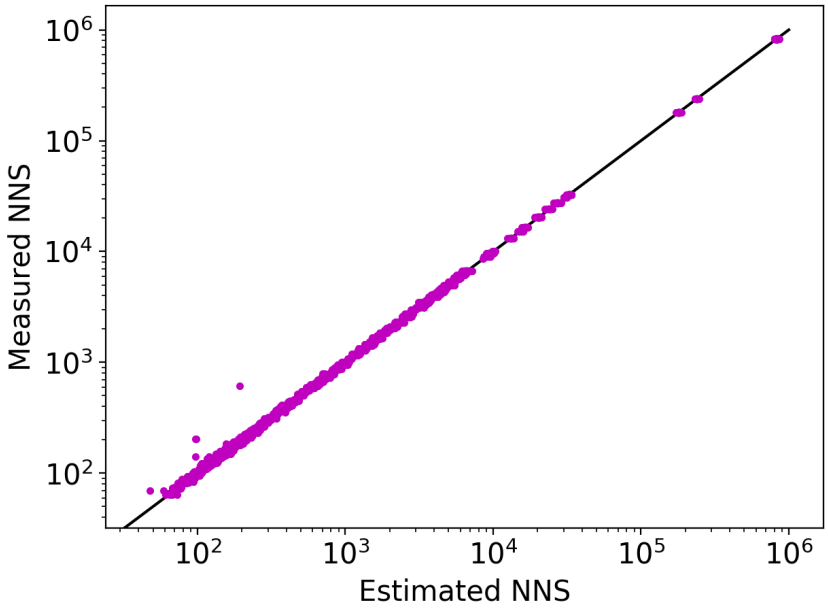
In this chapter, we have used Boolean threshold networks to study gene regulatory networks as GP maps. We defined genotype as the topology of the network, represented by its adjacency matrix, and defined its phenotype in three ways: as all the attractors in its state space, as the attractor occupying the largest part of its state space, and as a list of the attractors corresponding to all possible initial conditions of the network. We found that these phenotype definitions produce a series of properties observed in other GP maps in

Chapters 1 and 2, such as a very biased distribution of neutral network sizes, a linear scaling between phenotype robustness and the logarithm of phenotype frequency, positive correlation also between phenotype robustness and evolvability, and finally, simplicity bias. We have also found, for the GRN modelling the yeast cell-division cycle, that the neutral network of the wild-type phenotype is unusually large when compared to other phenotypes in this system, which suggests that this phenotype would be more likely to be found by random mutations.

Although these results are interesting, so far we have only explored the space of GRN topologies, having fixed the strength of gene-gene interactions to values of +1 and -1. One can naturally ask if we would obtain the same results if we had instead fixed the network topology and varied the strength of gene-gene interactions. This question is addressed in Chapter 4, where we move away from Boolean network models and propose a completely different GP map for gene networks.



(a)



(b)

Figure 3.18: **Neutral network sizes (NNS) for $N = 4$, for the phenotype defined as the list of attractors. (a)** Rank plot comparing NNS obtained by full enumeration of genotype space (shown in pink) to their results using the Monte Carlo NNSE described in this section (shown in grey), for the GP map using the list of attractors as its phenotype. The NNSE was run 20 times for each genotype, over 10^4 iterations for every run. **(b)** This plot shows the same data in **(a)**, with the estimated NNS in the x axis, versus the actual NNS on the y axis. The black line represents the identity function $y = x$.

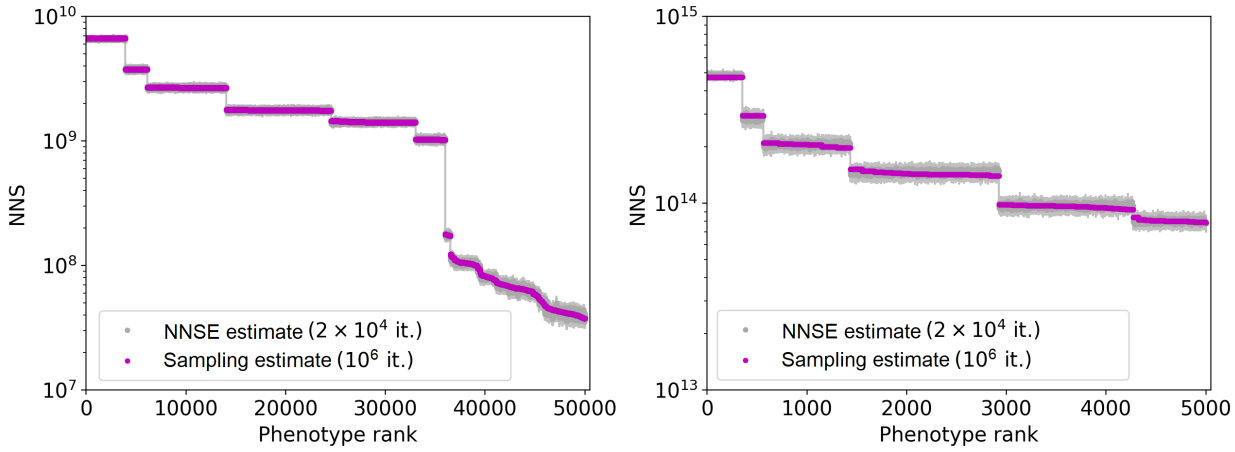


Figure 3.19: **Comparison between the results for (a) $N = 5$ and (b) $N = 6$ GRNs, for the phenotype defined as the list of attractors.** Both plots show phenotype neutral network sizes (NNS) obtained from sampling 10^6 genotypes (in pink) against results obtained with the NNSE algorithm, ran for 5×10^4 networks, over 2×10^4 iterations (in grey).

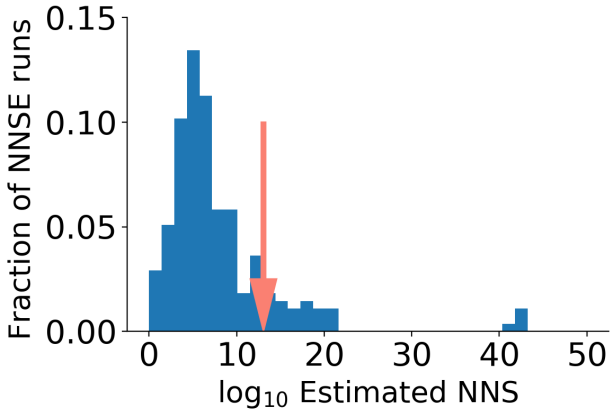


Figure 3.20: Results from NNSE algorithm for a sample of 200 genotypes with $N = 10$, run with `NTHERMAL`= 500 and `NRUNS`= 2000, for the phenotype defined as the list of attractors. The red arrow indicates the NNS of the fission yeast phenotype, which is larger than approximately 90% of all other neutral networks.

Chapter 4

Differential Equation Models for Gene Networks

"*These ambiguities, redundancies and deficiencies remind us of those which doctor Franz Kuhn attributes to a certain Chinese encyclopaedia entitled 'Celestial Empire of benevolent Knowledge'. In its remote pages it is written that the animals are divided into: (a) belonging to the emperor, (b) embalmed, (c) tame, (d) suckling pigs, (e) sirens, (f) fabulous ones, (g) stray dogs, (h) those included in this classification, (i) frenzied, (j) innumerable, (k) those drawn with a very fine camel hair brush, (l) et cetera, (m) having just broken a water pitcher, (n) those that from a long way off look like flies.*"

— J.L. Borges, on clustering methods. *El idioma analítico de John Wilkins*

4.1 Gene regulatory networks as differential equations

In the previous chapter, we studied GRNs through the lens of Boolean threshold networks, modelling the expression of a gene as the result of positive and negative regulation from other genes in the network. Alternatively, from a more mechanistic point of view, a gene network can be ultimately described as a set of interacting chemical reactions: a biochemical network. In modern-day systems biology, biochemical networks inside the cell are often modelled using differential equations, typically containing dozens of variables and sometimes over one hundred parameters. This framework was outlined in Chapter 1, where we mentioned how the expression pattern of a gene network can be described as a set of continuous variables $X_1(t), X_2(t), \ldots, X_M(t)$, following ordinary differential equations (ODEs) of the

form $dX_i/dt = f(X_1, \ldots, X_M)$, where f is typically composed of low-order polynomials and sigmoidal Hill functions [179, 224, 44]. This allows for more biologically realistic behaviour, which cannot be encoded in coarse-grained Boolean models such as the ones presented in Chapter 3.

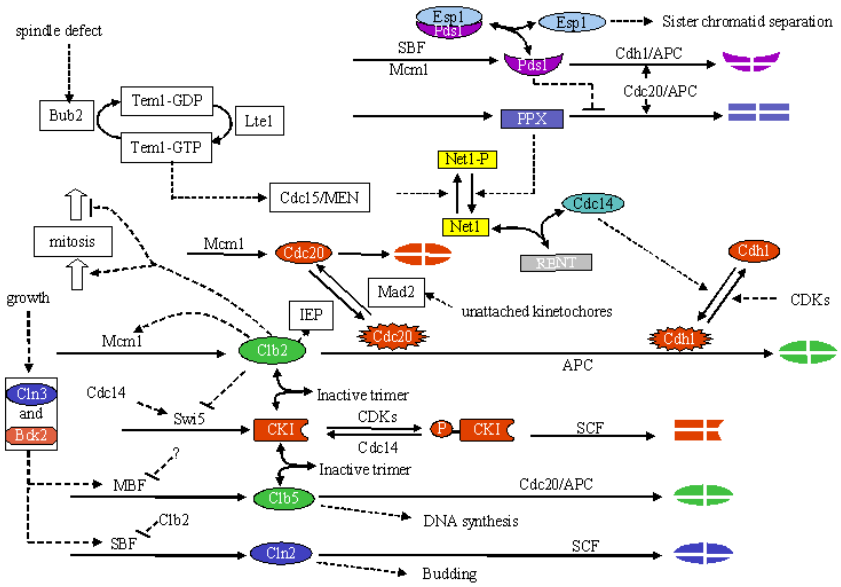


Figure 4.1: **Wiring diagram of the GRN controlling the budding yeast cell-division cycle, as described by Chen et al [42].** In this network diagram, nodes represent different molecules – cyclins, transcription factors, kinases etc – while arrows represent biochemical reactions such as phosphorylation, gene activation, and protein synthesis. The concentration of all molecules is governed by a set of 60 equations, relating 141 parameters and a set of 54 biochemical variables. The equations are presented in ref. [42].

Differential equation models have a long history of success in modelling cellular processes, being used to describe biochemical networks involved in the cell-division cycle [223, 253, 42], circadian rhythm [229, 137, 154], metabolic regulation [77], calcium oscillations [70], molecular motors [188], genetic oscillators [92], apoptosis [84], as well as many other processes [225]. Figure 4.1 shows one example of this class of models, from Chen et al's study of the budding yeast cell-division cycle [42], which maps the relation between 141 parameters and a set of 54 biochemical variables, which encode processes ranging from gene expression to mitotic spindle formation. The 60 differential equations describing this model are presented in ref. [42].

Models with so many parameters and variables carry with them the problem of *fitting* those parameters to experimental data: the size of the parameter space increases exponentially with the number of parameters of a model, and so the process of finding the right

parameters for a model becomes computationally expensive and potentially intractable. For example, even if every single one of the 141 parameters of Chen et al's model were to have only one of two possible values, this would imply an parameter space of $2^{141} \approx 10^{42}$ possible combinations – a space that could only be explored in hyperastronomical times. Yet, somehow, this ODE model has been very successful. For example, it has been used to quantiatively predict phenotype viability, sensitivity to biochemical parameters and mutant behaviour [42, 156, 104, 128].

Part of the explanation for how it might be possible to find parameters that reproduce experimental data, even when the parameter space is of a hyperastronomical size, comes from the notion of *sloppiness*, proposed in a series of papers by Jim Sethna's group [239, 102, 38]. Essentially, sloppiness is the degree to which the fit between model and data depends more on certain combinations of parameters than on others. This uneven dependency is seen in the Hessian matrix of the prediction error with respect to the parameters of the model, which presents a broad eigenvalue distribution, meaning that some eigenparameters affect the Hessian much more strongly than others [35, 34]. In other words, a small subset of the parameter combinations are stiff – small changes in parameters have large changes in the solutions – while most parameter combinations are sloppy – large changes in the parameter combinations have small effects on the solution. That said, sloppiness is a local property: it depends on a given set of measurements and on a given point in parameter space. A model might be more or less sloppy at different parts of parameter space, or with different measurements [220].

In this chapter, instead of coarse-graining the strength of gene-gene interactions to fixed strengths of activation/inhibition and only varying the topology of the network, which was done in Chapter 3, here we fix the network topology and vary the strength of each interaction, i.e. each parameter of the differential equation model. We treat the mapping from parameters to behaviours as a genotype-phenotype map, with sets of parameters as genotypes, and the time series resulting from the ODE model as phenotypes.

Of course, an important difference between these maps and the BTN maps of the previous chapter is that the parameters can take continuous values. So, as shown in Chapter 2 for the circadian rhythm mapping, we coarse-grain the inputs and the outputs to generate a discrete input-output map. For the inputs this discretisation is rather straightforward, but for the outputs (the phenotypes) this is less clear cut. This chapter will study two different

phenotype definitions, one based on the up-down method by Fink et al. [80, 242] on the time series corresponding to one molecule in the network, as well as an alternative definition based on the BIRCH clustering method [250]. Encouragingly, both definitions show the same global behaviour: a large variation in the neutral set sizes, a positive correlation between phenotype robustness and evolvability. Perhaps most interestingly, both phenotype definitions show that the wild-type parameters found in nature correspond to the largest of the neutral sets, much as was seen earlier by Dingle, Schaper and Louis for RNA secondary structures [68].

4.2 Defining the genotype-phenotype map

An ODE model of a GRN can be abstracted as a map from set of N parameters to a set of M curves over a continuous variable (typically time). Since an ODE model is a map from a continuous parameter space to a continuous behaviour space, it is necessary to coarse-grain both spaces before applying the discrete tools used in the previous chapters.

4.2.1 Discretising the input space

In genotype space, we approach this problem in the same way as done for the circadian rhythm GP map described in Chapter 2: for every ODE model we study in this chapter, we set all parameters to their wild-type values, and scale each parameter by a random factor $\zeta \in \{0.25, 0.50, \ldots, 1.75, 2.0\}$, chosen with uniform probability. This discrete scaling results in a discrete genotype space.

While this choice of discretisation of genotype space is arbitrary, it does not impact our results. This was discussed in Chapter 2, where we showed that one obtains the same results when discretising only the parameters of the ODE model, discretising parameters and initial conditions, or sampling parameters uniformly from their $\zeta \in [0.25, 2.0]$ range.

Once genotypes have been discretised, it is also necessary to coarse-grain the model outputs into biologically meaningful phenotypes. For instance, if two sets of parameters give rise to concentration curves that only differ slightly, they might best be counted as the same phenotype. In this chapter, we discretised phenotypes using two methods, described below.

4.2.2 Phenotype 1: up-down method

The first method we use to coarse-grain the outputs is the same one used in Chapter 2 for the circadian rhythm model in ref. [229], namely the "up-down" method by Fink et al [242, 80]. This method consists of taking an output curve $y(t)$ calculated in an interval $t \in [0, T]$, calculating its slope dy/dt at intervals of $t = \delta t, 2\delta t, 3\delta t, \ldots$, and printing the sign of dy/dt in every interval: for $j = 1, \ldots, T/\delta t$, if $dy/dt \geq 0$ (or < 0) at $t = j\delta t$, the j-th bit of the output string is written as a 1 (or a 0). The resulting string encodes the oscillations of $y(t)$ in a coarse-grained manner: curves with more oscillations will result in more complex strings, while curves with a smaller number of oscillations will result in strings with longer sequences of 0s and 1s.

Since a system of ODEs will produce one time series $y(t)$ for every variable in the model, each curve could give rise to a different coarse-grained binary string. Typically, the most biologically meaningful variable will be the biological output of the biochemical reaction network. This is not always precisely defined, but typically the most meaningful output molecules will be downstream, often at the bottom of the regulatory cascade, regulated by a large number of parameters in the network. This dependence on many parameters is what them the most relevant and informative description of the network's behaviour. For each ODE system, we then pick the "key molecule" that best represents the biological output of its network, coarse-grain its time series and take the resulting binary string as the network phenotype. The key molecules corresponding to the ODE systems studied here are listed in Table 4.1.

All ODE systems were simulated using a mix of custom Python scripts and the open-source modeling environment SloppyCell, version 1.0 [175, 101]. We sampled 5×10^6 genotypes, and discretised output curves in 50 time bins. The robustness of our results with respect to these implementation choices is discussed in Appendix C.

4.2.3 Phenotype 2: clusters of time series

The second method we use to define phenotypes uses cluster analysis. The output of an ODE system with m variables over time, when discretised in n time bins, becomes an mn-tuple of real positive numbers. If part of these mn-dimensional points lie close to each other, according to some choice of metric, they can then be clustered as the same phenotype, and the number genotypes corresponding to the same phenotype can be defined as the number

of points falling into each cluster. One can then use different *clustering methods* to assign each ODE output to its corresponding cluster, and since every genotype produces one ODE output, each genotype is then also associated to a cluster.

The choice of clustering method depends on the dataset to be clustered. For our data – mn-tuples representing time series coming from the ODEs – it is important to choose an algorithm that can deal with large numbers of data points in high dimensions, as there will be one mn-dimensional data point per genotype. This rules out algorithms such as Affinity Propagation, which is $\mathcal{O}(N^2)$, where N is the number of data points [191]. A priori, these clusters might also have different size and density of points, which would lead to poor performance of algorithms such as k-means and DBSCAN. For a detailed and practical description of the advantages and disadvantages of different clustering algorithms on different datasets, we point the reader to the documentation of the Scikit-learn Python machine learning library [185].

In this work, we use the BIRCH algorithm, standing for *Balanced Iterative Reducing and Clustering using Hierarchies* [250]. This algorithm is suited for clusters of different density and size, and is often used with large datasets, due to its good performance even with limited memory and computational time complexity of $\mathcal{O}(N)$ on the number of data points [191, 158]. BIRCH has also already been used in the context of GP maps: Raman and Wagner used it to cluster the concentration trajectories of signalling molecules, in a genotype-phenotype study of the *Saccharomyces cerevisae* target-of-rapamycin signalling circuit [193].

When applying a clustering method, it is fundamental to choose a distance metric that appropriately represents the difference between phenotypes. For instance, metrics based on Euclidean distance will highlight point-point differences between the ODE output curves, possibly returning high distance values for curves which differ only by a phase shift. On the other hand, Fourier-based methods will not count phase shifting differences, but might put too much weight on differences due to aliasing or to numerical errors when solving the ODEs. Besides phase shifting and aliasing differences, there is also the problem that different concentration curves might differ in orders of magnitude, as some molecules might have much higher cellular concentrations than others. This raises multiple methodological choices: should each curve be normalised by its maximum value, so they all range from 0 to 1? Or should amplitude be taken in consideration? And how important are these

questions, that is, how often do curves actually differ by scaling or phase shifting? We discuss these questions in Appendix C, where we test multiple combinations of metric and output representation.

In the same way as for the up-down phenotype, we sampled 5×10^6 genotypes, simulated ODE systems using custom Python scripts and the SloppyCell software [175, 101], and discretised output curves in 50 time bins. Alternative implementation choices are also discussed in Appendix C. Time series were clustered using the implementation of the BIRCH algorithm present in the Scikit-learn Python machine learning library [185], using the Euclidean distance as the clustering metric. Figure 4.2 shows examples of two phenotype clusters for the cell-division cycle GRN model by Tyson et al [223].

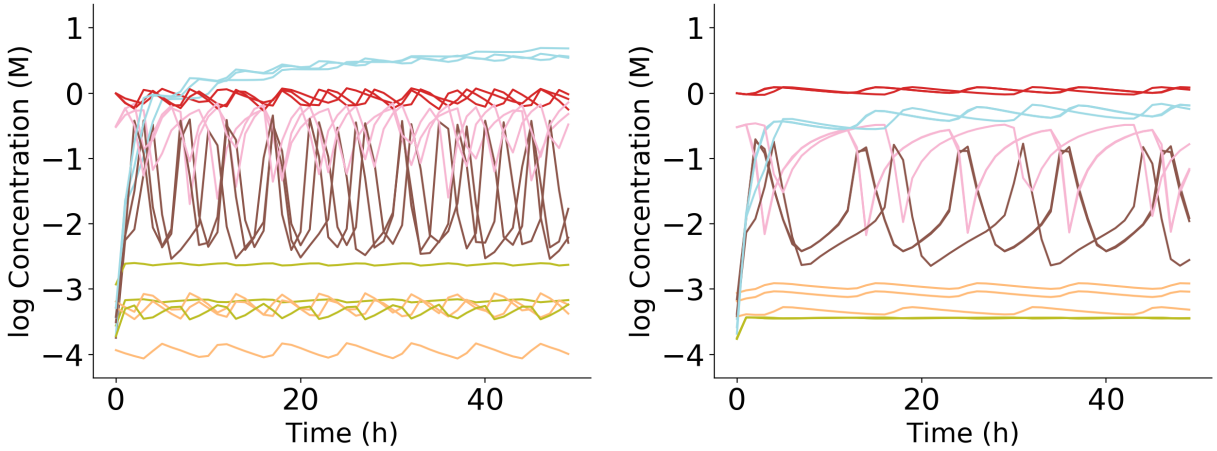


Figure 4.2: Examples of phenotype clusters for the cell-division cycle GRN model by Tyson et al [223]. **(a)** and **(b)** show two different phenotype clusters, presenting three output curves belonging in each cluster. Different colours indicate different biomolecules in the GRN.

Finally, it is important to verify that the clustering method is indeed separating phenotypes in disjoint clusters. Following Raman and Wagner's approach in ref. [193], we compare the distances between phenotypes in the same cluster, i.e. intra-cluster distances, to distances between phenotypes in different clusters, i.e. intercluster distances, for all clusters in each model.

4.2.4 GRNs studied in this chapter

In this study, we used a set of 13 ODE models of biochemical networks, including gene regulatory networks, but also signalling and metabolic networks. All models were taken

from a meta-analysis of sloppiness in different ODE systems by Gutenkunst et al [102]. The ODE models we chose are listed in Table 4.1, and describe biological systems ranging from rat growth-factor signaling [34, 201] to *E. coli* carbon metabolism [40] and circadian rhythm in the *Arabidopsis* plant [154].

Model name	Model system	Key molecule	Ref.
Brown 2004	Rat growth-factor signaling	Active form of B-Raf	[34]
Chassagnole 2002	*E. coli* carbon metabolism	2-Phosphoglycerate	[40]
Chen 2004	Budding yeast cell-division cycle	CLB2/SIC1 complex	[42]
Edelstein 1996	Nicotinic acetylcholine receptor	Biliganded acetylcholine	[72]
Kholodenko 2000	Protein kinase cascade	MAP2K	[123]
Lee 2003	*Xenopus* Wnt signalling pathway	Non-active beta-catenin	[136]
Leloup 1999	*Drosophila* circadian rhythm	Nuclear PER-TIM	[137]
Locke 2005	*Arabidopsis* circadian rhythm	X protein in nucleus	[154]
Sasagawa 2005	Rat growth-factor signaling	Ras-GDP complex	[201]
Tyson 1991	Eukaryotic cell-division cycle	P-Cyclin-Cdc2-P	[223]
Ueda 2001	*Drosophila* circadian rhythm	Cytoplasmic Clk-Cyc	[226]
Vilar 2002	Generic circadian rhythm	C Protein	[229]
Zak 2003	*In silico* regulatory network	Gene A	[245]

Table 4.1: List of all the differential equation models for gene regulatory networks used in this chapter, presented originally in the meta-analysis by Gutenkunst et al. [102]. The key molecule indicated for every model is the biologically relevant output used in phenotype 1.

4.3 Distribution of NNS for phenotype 1

As mentioned above, the GP map from a set of ODE parameters of a system of ODEs to a binary string representing one of the output variables of the gene regulatory network had already been explored in Chapter 2, for Vilar et al's circadian rhythm model [229]. Figure 4.3 shows the result of sampling 5×10^6 genotypes for 12 ODE models studied in this chapter, including Vilar et al's model. Although the number of phenotypes varies among maps, they all show a wide range of neutral network sizes, with a few phenotypes taking over the majority of genotype space, similar to what was observed for other GP maps in the previous chapters.

Perhaps the most remarkable result presented in Figure 4.3 is the frequency of the wild-type phenotype in comparison to other phenotypes. For all maps studied in this chapter,

the wild-type phenotype has an unusually large neutral network, often being the largest neutral network in the whole GP map. Taking in consideration the bias in the distribution of neutral network sizes, this means that the frequency of all wild-type phenotypes is orders of magnitude the frequency of most other phenotypes. Note that while Figures 4.3b, 4.3e and 4.3h, might seem to suggest that the wild-type phenotype has a low frequency, one has to remember that both axes of the figure are in logscale: even for these GRNs, the wild-type is still more frequent than a fraction of over 85% of all other phenotypes. These fractions are listed in Figure 4.3.

The fact that wild-type phenotypes are orders of magnitude more frequent than the average phenotype might have important evolutionary implications. It indicates that these phenotypes might be more easily found by random mutations than any random phenotype, and also suggests that the arrival of the frequent effect might be at play in the GRN parameters-to-behaviour GP map, in the same way as for the RNA map discussed in Chapters 1 and 2. In other words, these results suggest that the biased structure of this GP map might have constrained the phenotypic variation of GRNs available to natural selection, resulting in the very designable wild-type phenotypes we observe.

Apart from its evolutionary implications, the structure of this GP map also sheds a light on how it might be possible to find sets of over a hundred parameters for ODE models such as Chen et al's model of the yeast cell-division cycle [42], illustrated in Figure 4.1. Since fitting a model is not a task of finding a specific parameter set, but rather of finding *any* parameter set that produces a specific behaviour, this task becomes much easier if the target wild-type behaviour is highly designable, i.e. if many parameter combinations result in that phenotype.

In addition to the skewed distribution of neutral network sizes, these GP maps also present the same simplicity bias which was observed in other GP maps in Chapters 2 and 3. This can be seen in Figure 4.4, which shows plots of phenotype probability versus complexity for the same binary string phenotypes and the same GRNs presented in Figure 4.3, with all phenotypes with the same probability and complexity plotted on the same $(P(x), \tilde{K}(x))$ dot. All these GRNs show simplicity bias, which suggests that this behaviour does not depend on the biological specifics of any given gene network. Instead, it appears to be a general property of all these maps from GRN parameters to strings encoding their phenotypes, which may reflect a common feature of GP maps.

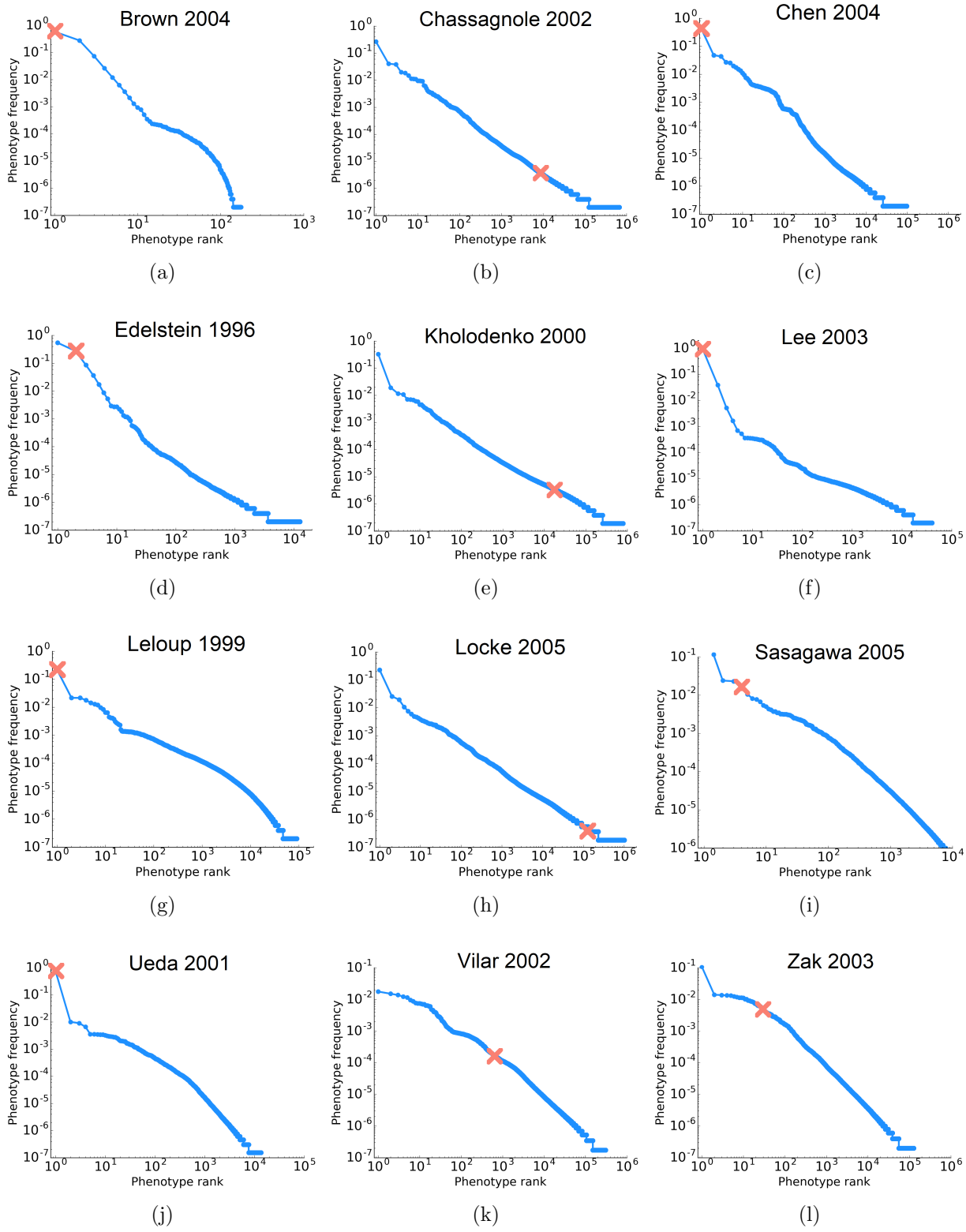


Figure 4.3: **Distributions of phenotype frequency for 12 ODE models of GRNs, for phenotype 1.** For all models, the wild type has one of the largest frequencies, being more frequent than the following percentages of all phenotypes: **(a)** 100% [34] **(b)** 98.71% [40] **(c)** 100% [42] **(d)** 99.99% [72] **(e)** 97.86% [123] **(f)** 100% [136] **(g)** 100% [137] **(h)** 87.16% [154] **(i)** 99.98% [201] **(j)** 100% [226] **(k)** 99.78% [229] **(l)** 99.98% [245].

Not only are the wild-type phenotypes unusually frequent, but they are also often among the simplest phenotypes, as indicated by the red crosses in Figure 4.4. This result suggests that the arrival of the frequent, described above, is also the arrival of the *simplest*: due to the simplicity bias in this GP map, the range of phenotypes available to natural selection are constrained to a set mostly made of simple phenotypes.

That said, the wild-type is not always the simplest phenotype. For example, in Figures 4.4b, 4.4e, 4.4h and 4.4k, the wild-type phenotype is located closer to the midpoint of the range of complexity values. In other words, while the wild-type phenotype is on average much simpler and more frequent than the whole set of possible phenotypes for a given GRN, it is often not the most frequent or the most simple. The reason may be that the ODE model produces phenotypes that do not show the necessary biological behaviour. For instance, for circadian rhythm models such as the ones in Figures 4.3h and 4.3k, the ODE outputs of lowest complexity often are non-repeating patterns. Since a non-periodic circadian rhythm may not be biologically viable, these phenotypes may never be observed in nature. One possibility would be that even though the wild-type is not the most frequent phenotype, it might be the most frequent of all biologically viable phenotypes. This could also be the case for the *E.coli* carbon metabolism and protein kinase cascade models presented in Figures 4.3b and 4.3e respectively: perhaps the simplest behaviours of these ODE systems do not satisfy some biological constraint, resulting in non-viable phenotypes. In fact, for most maps, the wild-type phenotype is not the simplest phenotype. Presumably, the main reason is that the simpler ones don't solve the biological problem at hand. More work is needed to confirm this hypothesis, nevertheless the overall trend is clear: this GP map shows simplicity bias, and wild-type phenotypes are often among the simplest, most frequent phenotypes.

4.4 Distribution of NNS for phenotype 2

In addition to using the up-down method on one key output of the GRN, we also used the BIRCH algorithm to define phenotypes as clusters of the time-series of all the relevant variables in an ODE GRN model. Using this definition, which we call phenotype 2, one can also measure the neutral set size as the number of genotypes that fall in each cluster. Using this second definition of phenotype, we once again find a very biased distribution of neutral

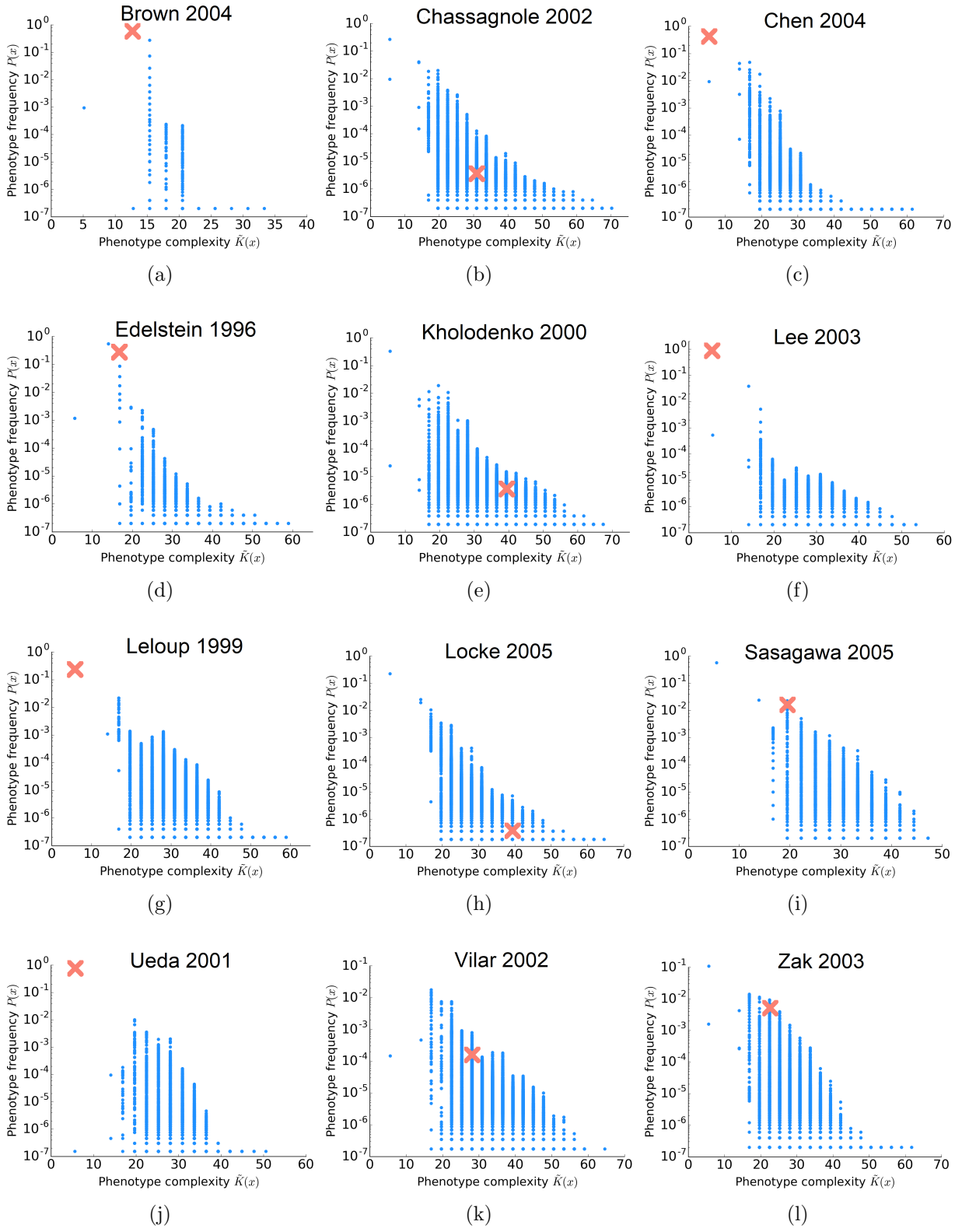


Figure 4.4: **Phenotype frequency versus phenotype complexity for 12 ODE models of GRNs, for phenotype 1.** All GRNs show the simplicity bias behaviour described in Chapter 2. The wild-type phenotype, marked with a red cross, is often among the simplest and most frequent phenotypes.

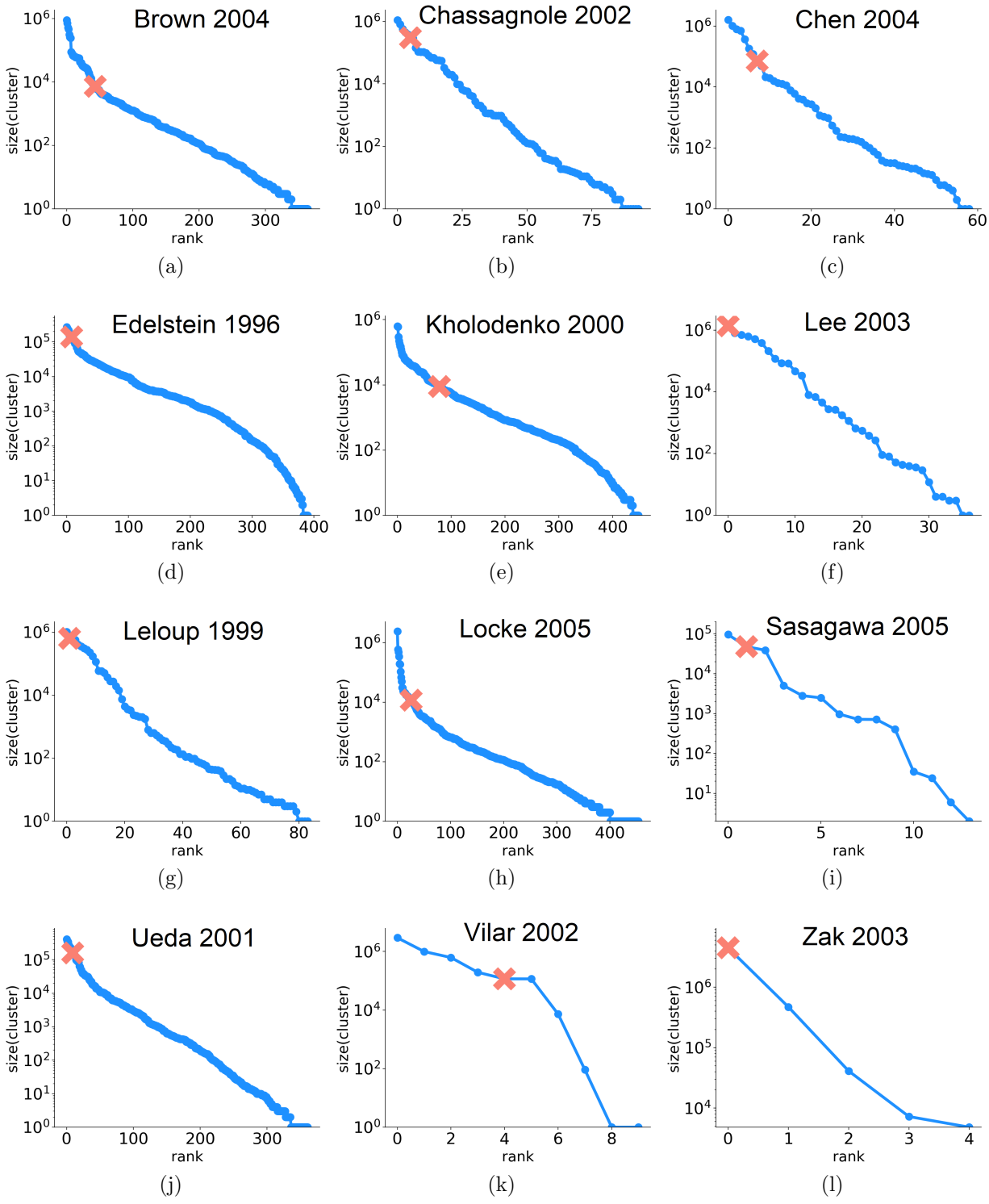


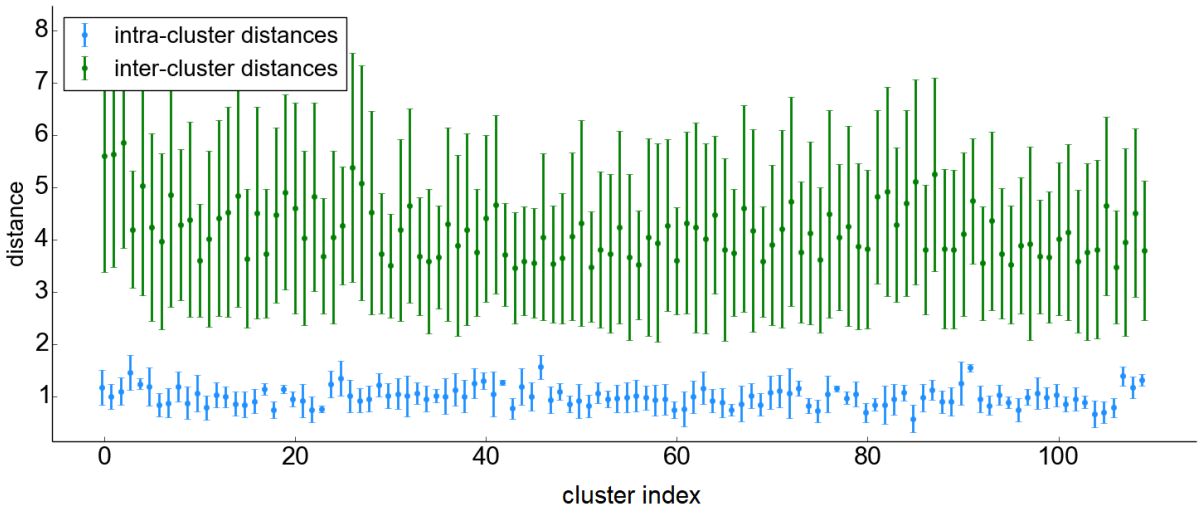
Figure 4.5: **Distribution of phenotype neutral set sizes for 12 ODE models of GRNs, for phenotype 2.** For all models, the wild-type phenotype (marked with a red cross) has one of the largest frequencies, being more frequent than the following percentages of all phenotypes: **(a)** 98.12% [34] **(b)** 94.69% [40] **(c)** 88.14% [42] **(d)** 97.95% [72] **(e)** 82.63% [123] **(f)** 100% [136] **(g)** 98.81% [137] **(h)** 94.29% [154] **(i)** 92.86% [201] **(j)** 97.51% [226] **(k)** 60% [229] **(l)** 100% [245].

network sizes. This is shown in Figure 4.5, for the same samples of 5×10^6 genotypes. We also confirm that, as observed for the binary string phenotype, the wild-type phenotype in all GRNs has one of the largest neutral networks, often being within one order of magnitude or less of the largest neutral network.

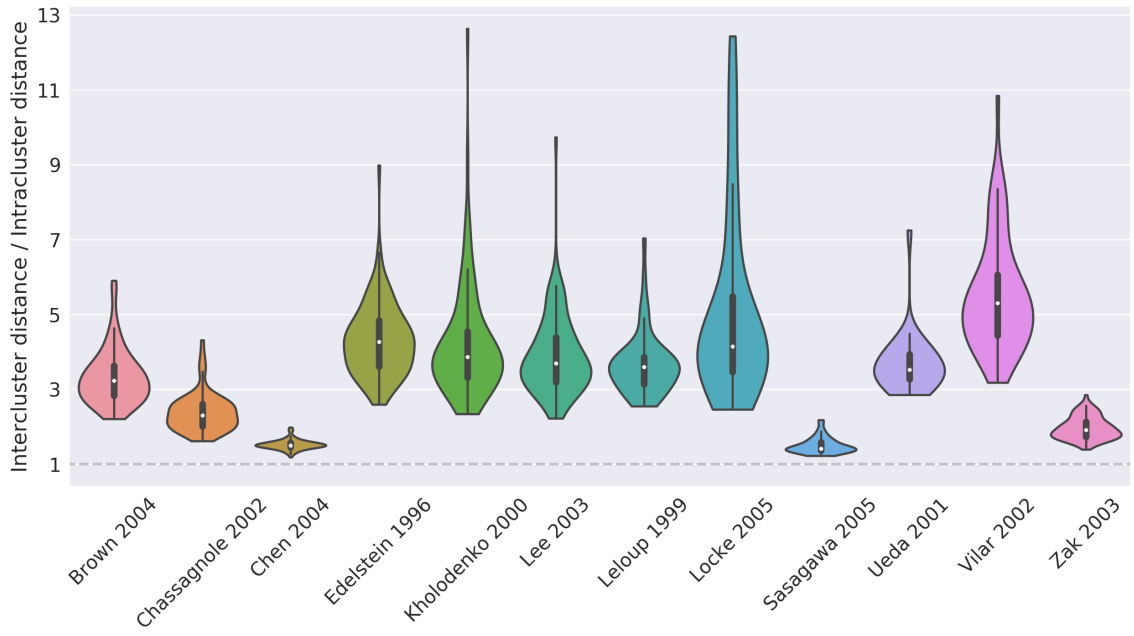
While the number of sampled genotypes was the same over all GRNs, the number of resulting phenotypes varies from 5 to a couple of hundred phenotypes. It is important to remember that this result comes from a sample of a small fraction of genotype space, which might affect the number of observed phenotypes: the GP maps where our sampling only resulted in a few phenotypes might actually have a larger range of phenotypes, but any phenotype with a neutral network occupying much less than 10^{-6} of genotype space is likely to not be present in a sample of 5×10^6 genotypes.

Finally, in Figure 4.6 we show the comparison of intra-cluster and inter-cluster phenotype distances. Figure 4.6a presents the result of this comparison for a single model, namely the nicotinic acetylcholine receptor dynamics GRN by Edelstein et al. [72], confirming that intercluster distances are always greater than intracluster distances.

All other GRNs in this study show the same pattern. The results for all GRNs are summarised in Figure 4.6b, where each violin plot represents one GRN model. In this figure, each violin represents the distribution of the ratio d_j^{out}/d_j^{in}. The numerator d_j^{out} represents the mean distance between phenotypes in cluster j and the midpoints of other clusters, resulting in a mean distance between cluster i and other clusters for that GRN. The denominator d_j^{in} represents the mean distance between pairs of phenotypes within cluster j. From Figure 4.6b, it is clear that this intercluster/intracluster distance ratio is always greater than 1, implying that the average pair of phenotypes within the same cluster is reliably more similar than phenotypes in different clusters. In other words, the clusters identified by the BIRCH algorithm are indeed separated from each other. In Appendix C, we validate our result against parameters of the BIRCH algorithm, as well as other choices of distance metric, output representation and level of discretisation of the ODE output time series.



(a)



(b)

Figure 4.6: **Comparison of intracluster and intercluster distances, for phenotype 2. (a)** Intra- and intercluster distances for all clusters for one GRN – in this example, the nicotinic acetylcholine receptor dynamics GRN by Edelstein et al. [72]. For all clusters, the distance between two outputs in the same cluster (intracluster distance) is consistently smaller than the distance between clusters (intercluster distance). **(b)** Distribution of the ratio between the mean intercluster distance and the mean intracluster distance for all clusters, plotted for the same 12 gene networks presented in the previous figures. All ratios are always above 1, indicating that the inter-cluster distance is always greater than the intracluster distance. From left to right, in different colours, the violin plots correspond to refs. [34], [40], [42], [72], [123], [136], [137], [154], [201], [226], [229] and [245].

4.5 Summary and discussion

In this chapter, we have shown that two very different definitions of phenotype lead to the same conclusions about this GP map for GRNs. We described the GP map from the parameters that describe the strength of gene-gene interactions to GRN behaviour, first defining phenotype 1 as an up-down string describing the oscillations of one key biomolecule over time, and also defining phenotype 2 as a cluster of time series, taking all molecules in consideration. We showed that, for both phenotype definitions, this map is biased, producing a wide range of neutral network sizes, that it has simplicity bias, and that the wild-type phenotypes have unusually large neutral networks, and often have very low complexity as well.

Overall, the GP map from a GRN to its gene expression behaviour is more ambiguous then self-assembly maps such as the RNA map, where the phenotype is clearly defined as the molecular secondary structure. To address this ambiguity, throughout Chapters 3 and 4, we defined phenotypes in various ways, such as sets of attractors in a Boolean state space, up-down discretisations of molecular concentration time series, and clusters of time series. Given that such different phenotype definitions all reveal the same structural properties, we believe these properties are indeed a property of GRNs, and not an artefact of any specific implementation of this GP map. These properties might also apply to signalling and metabolic networks, as they were also included in the models we studied in this chapter.

Considering that all the ODE models used in this chapter have very large numbers of parameters, resulting in high-dimensional parameter spaces, *a priori* the problem of finding parameter sets that fit the wild-type phenotype might seem untractable. Our results provide an explanation for how fitting these models might not be so difficult after all. The large neutral sets observed for wild-type phenotypes are essentially large sets of parameters that map to the same behaviour. This means that, given a GRN with the right topology, it is much easier to generate the wild-type behaviour than one might naively expect. These wild-type neutral sets are also known to be sloppy [102], meaning that not only are they large, but they are also non-isotropic; some directions in parameter space are more likely to produce phenotype change than others. What we find is that in addition to being sloppy, wild-type neutral networks are unusually large, when compared to other phenotypes for the same GRN.

Unlike the results concerning sloppiness described above, the findings we present here are global: they are not restricted to specific regions in parameter space, or to any particular measurement. Besides, while studies of sloppiness typically concentrate on the wild-type phenotype [102], in our work we compare the wild-type with other phenotypes for the same GRN, and find that the number of genotypes mapping to every phenotype is unevenly distributed, reflecting the other GP maps discussed in the previous chapters.

These findings might have important consequences for evolution. Considering that there is evidence that phenotypes with large neutral networks are might be more easily found by random mutations, as in the arrival of the frequent effect [202], and more likely to be observed in nature, as it is seen for the RNA GP map [68], our evidence suggests that the variation in GRN phenotypes observed in nature may be constrained to a set of frequent and low-complexity phenotypes. Natural selection would then operate within this constrained set.

That said, it is important to note that GRNs do not evolve simply by varying the strength of gene-gene interactions for a fixed GRN topology. Rather, they typically evolve by adding or subtracting components, thus changing the topology of the network. GRNs of varying topology were studied with the simpler BTN models in Chapter 3, although there we also restricted the number of genes to a fixed N. A more complete study of the evolution of GRNs would need to take both topology and parameters into account. Currently, such a model may be prohibitively complicated to implement.

Nevertheless, the more general AIT-derived results from Chapter 2 suggest that a very wide range of maps should exhibit phenomena such as simplicity bias, which in turn suggests that general models of GRN evolution may still exhibit the basic effects we observed in this chapter and in Chapter 3. In the next chapter, we will discuss another AIT-based effect that might constrain phenotypic variation in GRNs, and more generally in GP maps.

Chapter 5

Randomness Deficiency in Input-Output Maps

"*The art of simplicity is a puzzle of complexity.*"

— Douglas Horton

5.1 Simplicity beyond simplicity bias

In the previous chapters, we showed that many computable input-output maps exhibit simplicity bias, meaning that the number of inputs corresponding to each output is bounded by an exponential decay in the output's Kolmogorov complexity. In this chapter, we argue that simplicity bias alone does not account for all the simplicity observed in GP maps, or more generally in input-output maps. We propose that the notion of *randomness deficiency*, defined later in the chapter, might fill this explanatory gap.

As an example of simplicity which cannot be explained by simplicity bias, consider the RNA sequence-to-structure GP map, described in Chapters 1 and 2. The shape of a RNA molecule with n bases can be described (with some loss of information) by its secondary structure, which can be represented as a dot-bracket string, and translated to a $L = 2n$ bitstring, in such a way that every bitstring represents a different phenotype.

By the pigeonhole principle described in subsection 2.2.2, most strings are complex, which suggests most *possible* RNA secondary structures are also complex. However, this is not a direct implication, since most well-formed strings representing possible RNA structures are a very special subset of strings. That said, since the RNA map shows simplicity bias, any high-complexity phenotypes should correspond to fewer RNA sequences. The interplay

between these two effects should be seen in the range of phenotype complexity produced by this map: it might produce many complex phenotypes, but each complex phenotype might only correspond to a few RNA sequences.

We can test these hypothesis with $L = 20$ RNA, which is short enough that its $4^{20} \approx 10^{12}$ genotypes can still be fully enumerated, thus covering all possible outputs produced by this map [202]. The average c_{LZ} complexity of RNA structures, weighing all outputs equally, is of $\langle \tilde{K}(x) \rangle_{\text{RNA}} = 36.88$ bits. This is significantly lower than the average complexity of $\langle \tilde{K}(x) \rangle_{\text{well-formed}} = 46.08$ bits obtained from averaging over all well-formed $L = 20$ dot-bracket strings (where parentheses match), which is closer to the value of $\langle \tilde{K}(x) \rangle_{\text{all}} = 47.23$ bits over all the $n = 40$ bitstrings which can be produced from $L = 20$ dot-bracket strings. This discrepancy between the complexity of RNA structures and the complexity of random (well-formed or not) dot-bracket strings begs the question: where are all the complex shapes?

The matrix map, introduced in Chapter 2, shows similar behaviour to the RNA map. In Figures 2.13c and 2.13d, we show a series of violin plots llustrating how the average output complexity changes with the complexity of the row that defines the circulant matrix. Figure 2.13d indicates that low-complexity matrix maps show simplicity bias, and Figure 2.13c shows that the average output complexity $\tilde{K}_o$ is also lower for low-complexity matrix maps, regardless of how many inputs map to each output.

The low average output complexity for the RNA and simple matrix maps indicate that these maps produce a small, low-complexity subset of all possible outputs. By the pigeonhole principle, these low complexity strings are also rare. For example, both the RNA and the matrix map examples above have a mean complexity of outputs that is as much as 10 bits below the mean expected for random strings. According to the pigeonhole principle, strings as simple as these sum up to roughly $2^{-10} \approx 0.1\%$ of all strings of a given length. This result raises a question: why do these maps produce these rare low-complexity outputs?

Part of the answer must lie in the fact that the computable maps do not have the computational power of a UTM. Since UTMs can simulate any Turing machine on any input, by definition they can produce all possible outputs. That said, computable maps might still present a range of behaviours. In the next section, we separate computable input-output maps in four classes, depending on their complexity and number of outputs, and derive predictions for their average output complexity.

5.2 Classes of input-output maps

In this section, we present four classes of maps f mapping $\{0,1\}^m$ to a subset of $\{0,1\}^n$, and predict the average output complexity $\langle K(x)\rangle := \tilde{K}_o$ for each class of maps. The following derivations were produced in collaboration with Kamaludin Dingle and Guillermo Valle Pérez.

Limited complexity maps with few outputs

Given a map f, we previously defined f as having *limited complexity* when $K(f)+K(m) \ll K(x)+\mathcal{O}(1)$, where m is the size of the input strings. If a map f has limited complexity, and, simultaneously, the number of outputs N_O is much smaller than the total number of possible outputs of that length n, that is, $N_O \ll 2^n$, then we can associate an index $j \in \{1,\ldots,N_O\}$ to each string. With this indexing procedure, we can then bound the complexity of any output x:

$$K(x) \leq K(f) + K(m) + K(j) + \mathcal{O}(1) \tag{5.1}$$

where m is the size of the input strings, and j is the index of x in the list of outputs of the map f. The fact that $j \leq N_O$ it implies that $K(j) \leq \log_2(N_O)$, and therefore:

$$K(x) \lesssim K(f) + K(m) + \log_2(N_O) + \mathcal{O}(1) \tag{5.2}$$

Since, asymptotically, $K(f)$ and $K(m) \ll K(x)$, we can simplify the bound in equation (5.2):

$$K(x) \leq \log_2(N_O) + \mathcal{O}(1) \tag{5.3}$$

Also, according to the pigeonhole principle, most strings are complex, that is, $K(j) \approx \log_2(N_O)$ for most strings. In particular, this implies that the mean $\langle K(x)\rangle$ satisfies:

$$\langle K(x)\rangle \approx \log_2(N_O) \ll n \tag{5.4}$$

where it is again assumed that $N_O \ll 2^n$.

Limited complexity maps with many outputs

If the map f has limited complexity, but its number of outputs satisfies $N_O \approx 2^n$, most outputs will be complex. This is simply because nearly all outputs in $\{0,1\}^n$ are complex, which implies $\langle K(x)\rangle \approx n$.

Fully random maps

For a random (i.e. maximally complex) map f, then random samples of inputs are likely to map to random outputs, that is, outputs with high complexity. Note that this is the case even if the number of outputs $N_O \ll 2^n$: regardless of the number of outputs produced by the map, since all outputs are random strings, $\langle K(x) \rangle \approx n$.

Medium complexity maps

We can also define a map f mapping $\{0,1\}^m$ to a subset of $\{0,1\}^n$, where the rule set for f is neither completely random nor simple. We define *medium complexity maps* as maps with complexity $n < K(f) \ll N_O \log_2 N_O$, such that there is not enough information in f to encode N_O independent random strings (which would require $N_O \log_2 N_O$ bits). The outputs for maps in this class can be complex, even if the number of outputs N_O is much less than 2^n – they just cannot be fully independent from each other.

An example of this map is the random matrix map, discussed above: for this map, $K(f) = \mathcal{O}(n^2) \gg K(x)$, since $K(x) \leq n$. Using the same argument as for the simple maps, one can estimate that $K(x)$ is bounded by $K(f) + \log_2(N_O) + \mathcal{O}(1)$, and since most strings are complex, and therefore close to the upper bound, this suggests that $\langle K(x) \rangle$ is close to its maximum value, that is, $\langle K(x) \rangle \approx n$. Still, since $K(f)$ for the random matrix map grows as $\mathcal{O}(n^2)$ and the average N_O grows roughly proportional to 1.8^n, as seen in Figure 5.1, one can conclude that $K(f) \ll N_O \log_2 N_O$, which means the map does not contain enough information to define N_O random uncorrelated outputs. This suggests that, even though $\langle K(x) \rangle \approx n$, the outputs of the random matrix map must be correlated.

In the next section, we will discuss which of these four classes better describes what is observed for various maps. We will also discuss whether parameters such as the output length n and the number of outputs N_O are fundamental to an input-output map, or whether they depend on details of every specific implementation of the map.

5.3 Measuring randomness deficiency

A simple measure of the simplicity of a map is to compare the average complexity $\langle K(x) \rangle$ of its outputs to the average complexity $\langle K(x) \rangle_{all}$ of random strings of the same length as its outputs. This was done for the RNA map in section 5.1. The difference between $\langle K(x) \rangle_{all}$

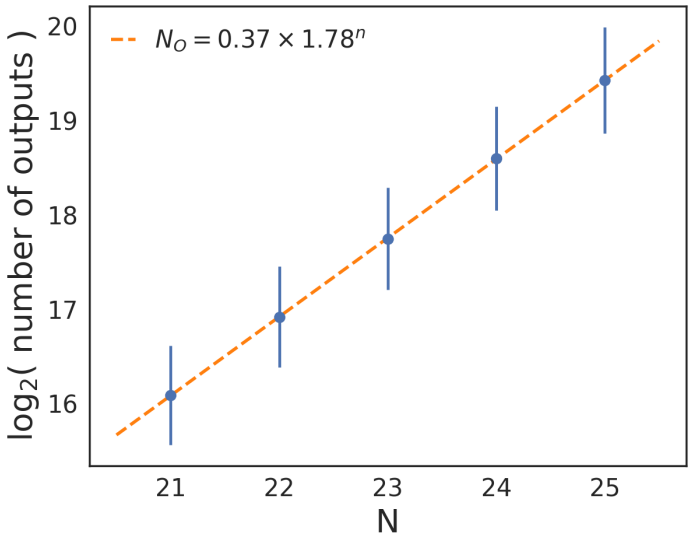


Figure 5.1: Mean and standard deviation of $\log_2(N_O)$, where N_O is the number of outputs of a random matrix map, for samples of 10^4 random matrix maps of size N ranging from 21 to 25, showing that the average number of outputs grows proportional to 1.78^n.

and $\langle K(x) \rangle$ can be linked to the concept of *randomness deficiency*, defined by Leonid Levin in ref. [143]. Levin defines the randomness deficiency of a string x with respect to a finite set S containing x as:

$$\delta(x|S) \; = \; \log_2 |S| - K(x|S), \tag{5.5}$$

where $|S|$ denotes the size of the set S. The randomness deficiency of x quantifies how typical it is within the set S. A randomness deficiency of 0 implies that specifying x requires as much information as specifying any typical random element of the set S, or in other words, that x is a typical object of S. On the other hand, a high randomness deficiency means that x is a rare element within S. Assuming that S is the whole space of n-bit strings, then $K(S) = n + O(1)$, and $K(x|S) = K(x|n) + O(1) \sim K(x)$, which means that $K(x|S)$ can be approximated by $K(x)$, and $|S| = 2^n$. We can then drop the indices, and write equation (5.5) as:

$$\delta(x) = n - K(x) \tag{5.6}$$

The randomness deficiency of a string x is also a measurement of how rare a string as complex as x is in among all 2^n strings, since the fraction of such strings is exactly $2^{-\delta(x)}$.

Measuring the average randomness deficiency of an input-output map would require measuring the average Kolmogorov complexity of its outputs. Since the Lempel-Ziv algorithm only returns an approximation of Kolmogorov complexity, we will use $\tilde{K}_0 := \langle \tilde{K}(x) \rangle_{all}$

instead of n. This way, the approximated randomness deficiency of a string x is defined as:

$$\tilde{\delta}(x) \;=\; \tilde{K}_0 - \tilde{K}(x) \tag{5.7}$$

We will drop the tildes for ease of notation. We will also represent the randomness deficiency of a map by $\langle\delta\rangle$, the average randomness deficiency of the outputs produced by a map.

Firstly, we confirm the prediction that the average output complexity should follow $\langle K(x)\rangle \approx \log_2(N_O)$ for limited complexity maps, stated in equation (5.4). Figure 5.2 shows the average output complexity $\langle K(x)\rangle$ plotted against $\log_2(N_O)$ for multiple input-output maps with output lengths $n = 20$. In purple, we show the RNA map, for $L = 10$ (i.e. $n = 20$), which shows low $\langle K(x)\rangle$ and low N_O. In red we show the Ornstein-Uhlenbeck map, which by definition has $\langle K(x)\rangle = K_0$, as well as $N_O = 2^n$ In blue, we show random matrix maps, which have narrow distributions of both $\log_2(N_O)$ and $\langle K(x)\rangle$, with $\langle K(x)\rangle \approx K_0$. In green, the we show the circulant matrix maps, which span over a wide range of both $\log_2(N_O)$ and $\langle K(x)\rangle$, revealing a linear relation between both variables, of the form $\langle K(x)\rangle \approx a\log_2(N_O)+b$, where $a \approx 1$. The offset $b \neq 0$ captures the $\mathcal{O}(1)$ terms introduced by the Lempel-Ziv algorithm. Note that the circulant matrix maps are a subset of the random matrix maps – but a very particular, low-complexity, rare subset.

Figure 5.2 also includes the random walk return map, shown in red and described in ref. [67]. Inputs for this map are defined as follows: A walk of m steps will produce a position vector $s = (s_1, s_2, \ldots, s_m)$, where s_j represents the position of the random walker at time j. Since each sequence of steps is equally likely, and will produce a unique path, we consider this path as an input. Outputs are defined as another sequence over m time steps, $x = (x_1, x_2, \ldots, x_m)$, where $x_j = 0$ if $s_j \leq 0$, and $x_j = 1$ otherwise. The changes from 0 to 1 (and vice-versa) in the output sequence thus correspond to times when the random walker returns to $s = 0$. This map can also be seen as a simplification of the Ornstein-Uhlenbeck map, when the mean-reverting parameter θ is set to zero. In this regime, all change in the price S_t will be due to the Brownian motion dW_t, which allows for steps of any size. In the random walk map, this motion is coarse-grained to a random walk with steps of fixed size. As the number of possible random walks with n steps is countable and finite, this allows the inputs to be fully enumerated. For more detail on the random walk return map, we direct the reader to ref. [67].

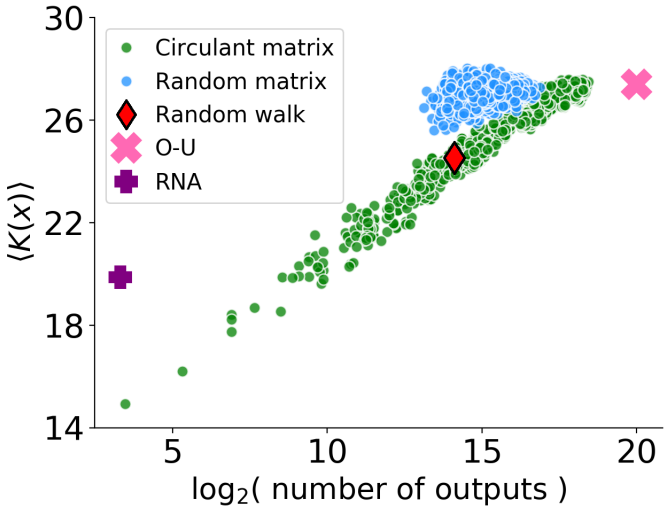
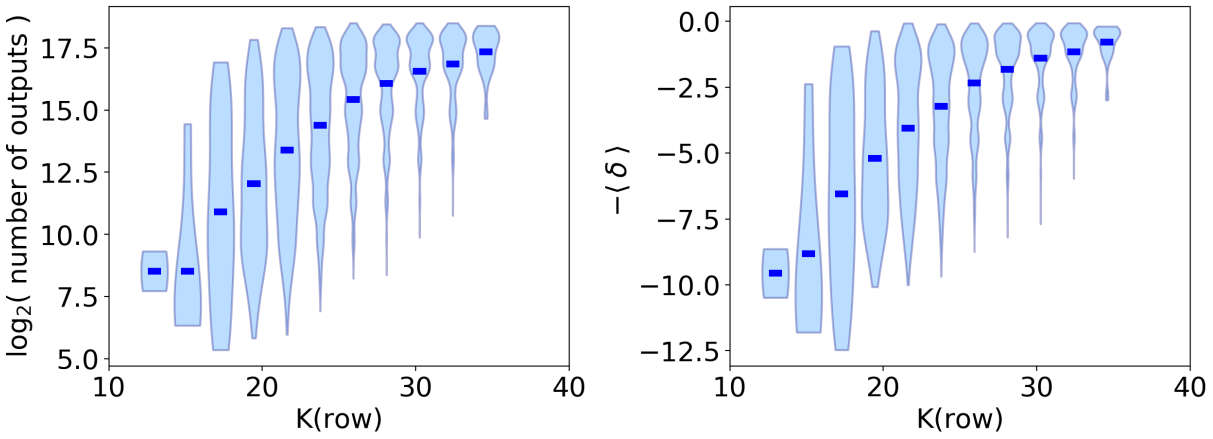


Figure 5.2: **Average output complexity ($\langle K(x)\rangle$) versus the logarithm of the number of outputs ($\log_2 N_O$) for different input-output maps.** Each colour represents a different map with $n = 20$ output strings. The random matrix maps, drawn in blue, show a narrow distribution with $\langle K(x)\rangle \approx K_0$, as well as large N_O. The circulant matrix maps, shown in green, cover a wide range of average output complexity and number of outputs, keeping a linear relationship of $\langle K(x)\rangle = \log_2 N_O + \mathcal{O}(1)$, as predicted in equation (5.4). The random walk map, shown as a red diamond, falls within the line drawn by the circulant matrix map. The Ornstein-Uhlenbeck map, shown as a pink $\times$, by definition has maximum N_O and $\langle K(x)\rangle$, and the RNA map, drawn as a purple $+$, shows low values for both variables.

In Figure 5.2, most variation in both $\log_2(N_O)$ and $\langle K(x)\rangle$ happens for the circulant matrix map. Figure 5.3 shows how these two quantities vary as a function of $K(\text{row})$, the complexity of the row that defines the circulant matrix. Since $\langle K(x)\rangle \approx a \log_2(N_O) + b$, it is not surprising that both Figures 5.3a and 5.3b result in very similar plots when compared with $K(\text{row})$. For the average randomness deficiency, instead of plotting $\langle\delta\rangle$, we plot $-\langle\delta\rangle$, or $\log_2(\, 2^{-\delta}\,)$, which is the fraction of the strings in $\{0,1\}^n$ which are as complex as x. In other words, $-\langle\delta\rangle \approx 0$ indicates a map producing complex outputs, and $-\langle\delta\rangle \ll 0$ indicates a map with high randomness deficit, i.e. a map that produces simple outputs.

Figure 5.4 shows measurements of randomness deficiency for the input-output maps presented in Figure 5.2, but for varying n. In the figure, blue violins represent 2.5×10^4 random matrix maps for each $n = 20$ to $n = 25$, green violins represent 2.5×10^4 circulant matrix maps for each $n = 20$ to $n = 25$, and red diamonds represent the random walk return map, for n in the same range. The pink $\times$ represents the $n = 20$ Ornstein-Uhlenbeck map, and the purple $+$ represents the $L = 10$ ($n = 20$) RNA map.

This figure confirms the values of $\langle K(x)\rangle$ for different classes of maps predicted in sec-



(a) (b)

Figure 5.3: **Randomness deficiency for the circulant matrix map.** This figure shows that **(a)** the number of outputs and **(b)** the average randomness deficiency $\langle\delta\rangle$ of the 20×20 circulant matrix map can be varied by changing the complexity $K(row)$ of the first row that defines the map. In these violin plots, made with 2.5×10^4 matrices, each violin shows the distribution of $\log_2(N_O)$ and $-\langle\delta\rangle$ in a standard violin plot format. The horizontal dark blue lines denote the mean value on the y axis for each value of $K(row)$.

tion 5.3. For the random matrix map, all blue violin plots lie close to the $\langle K(x)\rangle = K_0$ line, showing that the average output of a random matrix map is as random as a random string of same length. This means that random matrix maps show no randomness deficiency, which is expected for medium complexity maps. The same is true for the Ornstein-Uhlenbeck map, which by definition has $\langle K(x)\rangle = K_0$. The random walk return map, shows a small randomness deficiency, with a small but constant gap between $\langle K(x)\rangle$ and K_0. The RNA map and the circulant matrix maps, however, do show a significant degree of randomness deficiency, indicating that the outputs produced by these maps are less complex than random strings. For the average circulant matrix map, indicated by the horizontal green lines, and for the RNA map, $\langle\delta\rangle = K_0 - \langle K(x)\rangle \approx 10$. Considering that $2^{-\langle\delta\rangle} = 2^{-10} \approx 10^{-3}$, this means that one in every 1000 random strings is as simple as the average output produced by the circulant matrix map or by the $L = 10$ ($n = 20$) RNA map. For some circulant matrix maps, this value increases to $\langle\delta\rangle \approx 25$, meaning that only one in every $2^{25} \approx 3\times10^7$ random strings is as simple as the outputs produced by these maps.

From the plots above, it is clear that one cannot use the randomness deficiency δ to tell a random matrix map apart from a random map, since both maps show $\langle\delta\rangle \approx 0$. The same is true for the simplicity bias, which is absent in both maps. Still, we know that the matrix

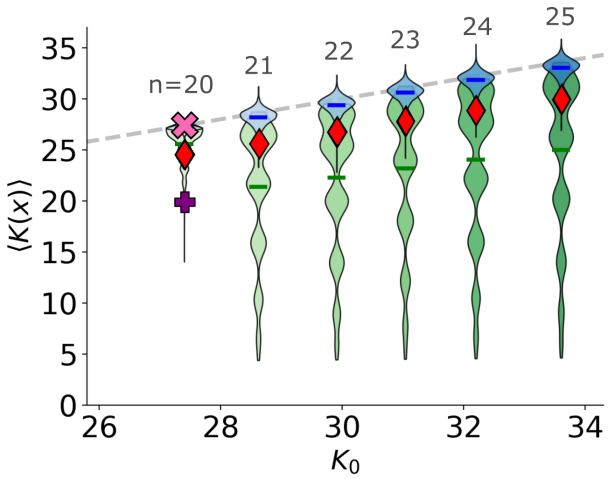


Figure 5.4: **Randomness deficiency shown for different maps and varying n.** This figure compares the average output complexity $\langle K(x)\rangle$ with K_0, for the same maps presented in Figure 5.2. For the random matrix map and the Ornstein-Uhlenbeck map, shown respectively as blue violin plots and as a pink $\times$, $\langle K(x)\rangle$ follows K_0 closely. For the random walk return map, shown as red diamonds, there is a gap between $\langle K(x)\rangle$ and K_0, indicating that the outputs produced by this map are less complex than random strings of the same length. For the RNA map, shown as a purple $+$, and for some circulant matrix maps, shown in green, this gap is much wider, indicating a stronger randomness deficiency. The violin plots represent samples of 2.5×10^4 random matrix maps and 2.5×10^4 circulant matrix maps for each $n = 20$ to 25. The random walk return map is also shown in this range. Horizontal bold lines represent the mean values of every violin plot, and the dashed line indicates the $\langle K(x)\rangle = K_0$ line.

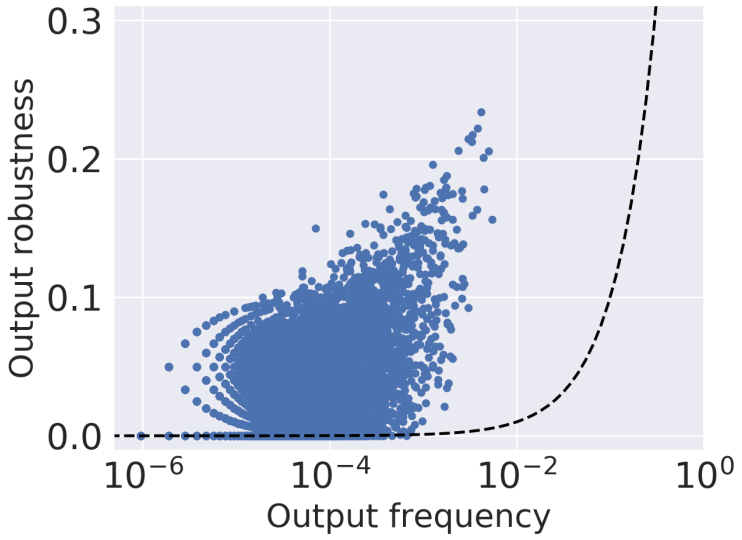


Figure 5.5: **The random matrix map is correlated.** This plot, produced for a single 20×20 random matrix map, compares the frequency of an output with its robustness, following the definition given in Chapter 1. The black line shows the output robustness expected for an map with no correlations between the outputs of similar inputs.

map is not random. As we show above, its complexity is less than the minimum complexity of $N_O \log_2 N_O$ which would be needed for a truly random map. Moreover, as shown in Chapter 2, even though the output spectrum of an average over many matrix maps is as complex as a sample of random strings, individual maps show bias towards specific strings. This non-uniform distribution of inputs over outputs is an effect which would not be present in a random map.

Given that the GP maps presented in Chapter 1 show correlations between inputs and outputs, it is interesting to see whether the random matrix map also shows this property. Figure 5.5 shows exactly that. This plot, produced for a single random matrix map, compares the frequency of an output with its robustness, defined in Chapter 1 as the average probability that a point mutation (in this case, a bit flip) on an input will not change its output. The random matrix map shows similar robustness behaviour as the GP maps discussed in Chapters 1 and 3, with output robustness scaling linearly with the logarithm of output frequency. The dashed black line indicates what would be expected for an map with no correlations between the outputs of similar inputs, which includes the random map. In summary, the random matrix map is much more correlated than a random map.

5.4 Discussion

In this chapter, we have explored a few classes of computable input-output maps, and showed that some of them exhibit randomness deficiency, that is, the outputs they produce are less complex than expected. In particular, we showed that limited complexity maps have an average output complexity of $\langle K(x) \rangle \approx \log_2(N_O)$, where N_O is the number of outputs of the map. For maps where $N_O \ll 2^n$, this implies that $\langle K(x) \rangle \ll n$. We also showed how the average output complexity and the number of outputs grow with the complexity of a circulant matrix map, or with the system size n. We also showed that the random matrix map, despite producing a complex output spectrum, is not as random as a random map, as it shows correlations where neighbour inputs are likely to map to the same outputs.

These results draw more parallels between artificial input-output maps such as the random matrix map and GP maps such as the ones presented in the previous chapters. In addition to simplicity bias, we have now shown that many of those maps also show randomness deficiency, probably due to constraints that limit the total number of outputs of a map,

consequently limiting their range of complexity. It is hard to say which effect is dominant, but it is evident that they work in tandem to produce simplicity in phenotypes, and more generally in outputs. In the next chapter, we will discuss these parallels between GP maps and other input-output maps in more detail.

Chapter 6

Conclusion

"*To conclude, we believe that there is a considerable gap in the neo-Darwinian theory of evolution, and we believe this gap to be of such a nature that it cannot be bridged within the current conception of biology.*"

— Marcel Schützenberger, *Algorithms and the neo-Darwinian theory of evolution*

6.1 Summary and discussion

In this thesis, we have studied a series of structural properties of genotype-phenotype (GP) maps, with a special focus on gene regulatory networks. We also investigated whether some of these properties are properties of more general input-output maps, and provided links between some of these structural properties and concepts from algorithmic information theory (AIT).

In Chapter 1, we presented a list of structural properties present in many GP maps in the literature. These properties include redundancy, meaning that the number of genotypes is larger than the number of phenotypes, also bias, referring to the biased distribution of the number of genotypes mapping to each phenotype, as well as a positive relationship between phenotype robustness and evolvability, following the formal definitions of robustness and evolvability in Andreas Wagner's work [233]. We discussed these properties in terms of *neutral networks*, a name given to the sets of genotypes that map to a given phenotype, as well as the neutral mutations that connect them. The number of genotypes in a neutral network is often called the neutral network size. We also use the term *frequency*, referring to the fraction of the whole genotype space taken by a given neutral network.

The large bias in how genotypes are distributed over phenotypes is known to play a big role in evolutionary dynamics, causing an effect known as the *arrival of the frequent*, described by Schaper and Louis [202]: when most genotypes map to only a few phenotypes with larger neutral networks, a random mutation is likely to result in one of those frequent phenotypes, and an evolving population subject to mutational drift is likely to move towards those phenotypes, even if they do not have the highest fitness of all the theoretically possible phenotypes. Naturally, a strong selection gradient can always steer a population away from those frequent phenotypes, but the wider the frequency gap between a frequent phenotype (larger neutral network) and a rare phenotype (smaller neutral network), the stronger the selection necessary to overcome the uneven availability of both phenotypes. For a GP map that shows a strongly biased distribution of neutral network sizes, one could expect that the phenotypes found in nature would be the phenotypes with the largest neutral networks.

This expectation turns out to be correct, at least for the GP map from RNA sequences to secondary structures. For this map, the RNA phenotypes present in biological databases have an unusually large neutral network size, typically orders of magnitude larger than the average phenotype neutral network for that GP map, which suggests that phenotypic variation in non-coding RNA might be strongly shaped by biases in the RNA sequence-to-structure map. So far, the RNA map is the only GP map where this effect has been observed [68].

The presence of structural properties in GP maps raises a series of questions. One could ask whether these patterns can be explained by any biological or physical principle, or whether these properties, observed so far mainly for GP maps of self-assembling systems, are actually more general properties of input-output maps.

Aiming to understand which kind of principles would hold for any sort of input-output map, in Chapter 2 we studied input-output maps using tools from AIT, which draws from Shannon information theory and from Turing computability theory. This powerful theoretical framework allowed us to show that, for a particular class of input-output maps, high frequency outputs are likely to be simple. This so-called *simplicity bias* is cast in terms of Kolmogorov complexity, which is a measure of the amount of information in an object (the output, in this case). For this property to hold, the main requirement is for the input-output map to be asymptotically simple, such that specifying the map and the size of its inputs requires less information than specifying an output. Formally, for a map f with inputs of length

n producing an output x, a limited complexity map satisfies $K(f) + K(n) \ll K(x) + \mathcal{O}(1)$. We then argued that many input-output maps in science and engineering can be cast as maps of limited complexity, which suggests that simplicity bias may be present in a very broad range of systems.

In Chapter 2, we also defined the matrix map, a very simple class of input-output maps with both inputs and outputs represented as binary strings, in such a way that it was possible to vary the complexity of the mapping - in other words, the amount of information necessary to build the map. We found that low-complexity matrix maps show simplicity bias, whereas with maps where $K(f) \geq \mathcal{O}(n)$ this effect seems to disappear.

In the following two chapters of this thesis, we applied ideas from Chapters 1 and 2 to another very broad category of GP maps, namely gene regulatory networks (GRNs). In both chapters, we defined the GP map as a map from the structure of a GRN to its gene expression patterns. In Chapter 3, we studied this GP map using Boolean threshold networks, which are a simplified model of GRNs. We studied the map from the topology of the network to the dynamical attractors of its gene expression patterns, defining the genotype as the adjacency matrix of the gene network, and the phenotype initially as the set of attractors in the state space, then as the attractor with the largest basin in state space, and then as a list of the attractors corresponding to each possible initial state of the network. We found that this map presents the structural properties of GP maps discussed in Chapter 1, such as a very skewed distribution of neutral network size, and a positive correlation between phenotype robustness and evolvability. We also showed that this map also produces simplicity bias, with the simplest outputs typically being considerably more frequent than their more complex counterparts, often by orders of magnitude.

Since the genotype spaces corresponding to biologically relevant GRNs are typically too large for normal sampling methods to be used, we adapted Jörg et al's NNSE algorithm [112] to estimate the neutral network size for GRNs, including the network corresponding to the wild-type phenotype of the fission yeast cell-division cycle studied by Davidich and Bornholdt [58]. We found that the frequency of the wild-type phenotype is orders of magnitude above the frequency of most phenotypes. This suggests that the phenotypic variation in GRNs may be constrained by the biases in the GP map, which might result in an arrival of the frequent effect similar to what is observed for the RNA map.

The formalism of Boolean threshold networks, while allowing the topology of a GRN to change, has the disadvantage of not allowing the gene-gene interactions to vary in strength. Likewise, this model reduces gene expression to a binary (Boolean) value. In Chapter 4, we addressed these problems, studying GRNs using systems of nonlinear ordinary differential equations (ODEs). This class of models has been historically very successful in studying GRNs and signalling transduction networks. ODEs are also in stark contrast with the Boolean networks mentioned above: for ODEs, all variables and parameters are continuous, and instead of fixing the strength of gene-gene interactions and varying the gene network topology, in this model we fixed the GRN topology and varied its interaction strengths. We studied a set of 12 ODE models studied by Gutenkunst et al. [102], describing gene networks in systems ranging from plant circadian rhythms [154] to bacterial carbon metabolism [40].

Given the continuous nature of ODE outputs, we took two approaches at defining phenotypes. First, we used the up-down method by Fink et al. [242, 80] on the concentration time series of a key biomolecule of each GRN. Alternatively, we took the time series of all molecules in the GRN, and used the BIRCH clustering algorithm [250] to identify clusters of similar outputs. We then defined phenotypes as clusters of outputs.

While conceptually different, both methods of defining the phenotype show a large bias in the distribution of neutral network size. Moreover, the wild-type phenotypes are typically among those with the largest neutral networks, reflecting what was found using Boolean threshold networks in Chapter 3. Even though the wild-type phenotype changes completely across organisms, we found that, in all 12 ODE models studied in Chapter 4, the phenotype observed in nature was always one of the most frequent phenotypes, as expected from the arrival of the frequent effect. Considering that the same qualitative behaviour is observed across organisms, and for different models of GRNs, we argue that this may hold much more generally for GRNs. Besides, since the ODE models studied in Chapter 4 also included signalling and metabolic networks, these properties might hold for these classes of biochemical networks as well.

Given the positive correlation between phenotype frequency and robustness, it appears that the latter comes "for free", as a by-product of the arrival of the frequent: since the genotype spaces where evolution happens are hyperastronomically large, evolutionary search is likely to only find the most frequent phenotypes, which also happen to be the ones with

the highest robustness. And given the correlation between robustness and evolvability, these phenotypes are also likely to be very evolvable.

Finding that the wild-type phenotypes have unusually large neutral networks also provides a solution for a modelling problem: as the number of parameter combinations grows exponentially with the number of parameters of the model, models with many parameters become notoriously hard to fit. In Chapter 4, we presented a few ODE models with over a hundred parameters [42, 201], which were all fit to experimental data. Considering that this GP map shows redundancy and bias, with the frequency of the wild-type phenotype often being orders of magnitude above the frequency of most other phenotypes, fitting a model stops being a matter of finding the ideal parameter set, and becomes a matter of finding one of the many parameter sets that map to the wild-type phenotype.

The reader might find similarities between our results and the concept of *sloppiness*, developed in a number of papers by Jim Sethna's group [239, 38]. In its essence, sloppiness is the degree to which the fit between model and data depends more on certain combinations of parameters than on others [35, 34]. Sethna's group has shown that, in models with many parameters, most parameters can vary over a wide range while having little effect on the fit between model and data. Those *sloppy* parameters (or degrees or freedom) are proposed as an explanation for how it might be possible to fit a model with hundreds of parameters: one would only need to fit the few combinations of parameters which actually affect the fit between model and data.

That said, the conclusions of the studies of sloppiness are quite different from our results. While sloppiness is a local property, describing the relation between one point in the model parameter space and a set of measurements, the results we present here are global, referring to neutral networks in the whole genotype space. While sloppiness studies normally focus on the wild-type phenotype [102], we describe statistical properties such as phenotype frequency, robustness, and evolvability for all phenotypes produced by a GP map. Moreover, while sloppiness describes the *shape* of the region in parameter space that maps to a given output, indicating which directions in parameter space have a stronger effect on the output, our results describe the *size* of that same region, i.e. its neutral set size. So while all the different outputs may present sloppiness, our results indicate that the volume of the set of parameters that produces a particular output varies over many orders of magnitude, depending on the output. In particular, for simpler outputs (phenotypes), this volume (neutral set

size) tends to be larger.

In Chapter 5, we went back to looking at general properties of input-output maps, and argued that simplicity bias is not sufficient to explain the range of complexities that are observed in nature. We then argued that in addition to simplicity bias, there is another source of simplicity which can be described by the AIT concept of *randomness deficiency*, which measures how different a set of outputs is from a null expectation of random strings. Using arguments from AIT, we predicted randomness deficiency for different classes of input-output maps, and tested our measurements against a series of maps presented in Chapter 2. In particular, we showed that input-output maps of limited complexity can have a high randomness deficiency, which implies that this property should be seen in any GP maps which also fall in this category, such as the GRN GP maps discussed in Chapters 3 and 4. More broadly, the combined action of randomness deficiency and simplicity bias may help explain the origins of simple, compressible patterns in nature, such as symmetrical shapes or periodic behaviour in biological systems.

6.2 Outlook: gene networks and evolution

There is much more to explore for gene networks. From Chapters 3 and 4, it is clear that there are many open questions about how to define genotype and phenotype for this GP map, although we are encouraged by different methods producing similar results. It is important to remember that the evolution of GRNs does not happen by only changing gene-gene interaction strengths, nor does it happen by only changing the topology of a gene network of fixed size. Rather, it often occurs through gain or loss of genes. Naturally, a more biologically realistic GP map which addressed the effects of changing GRN size, topology and interactions at the same time would probably be significantly more complex than the models studied in this thesis, but it is important to take this step, to further validate our results.

The results we found for GRNs are very general, concerning overall biases towards simplicity, in the form of short attractors in state space, or in the form of low-complexity time series outputs. Considering the diversity of intracellular processes governed by the GRNs discussed in this thesis, one is prompted to ask more biologically concrete questions, such as: what kind of functionality can be easily found (has high frequency) by this GP map?

Given that the structure of this GP map constrains its phenotypic variation, which kinds of biological oscillators, signalling cascades, and cell-division cycles are more likely to be found by evolution? Also, how can we see the effect of selection on top of these constraints? In particular, since some GRNs control embryonic development, what do these results imply about development? And more broadly, do any other GP maps show the arrival of the frequent, beyond RNA and GRNs?

Although there are numerous questions that can be asked regarding specific biases for specific GRNs, one of the biggest results in this thesis is the strong bias towards simplicity. This bias is shown by a wide range of biological and non-biological input-output maps, and seen through both simplicity bias and randomness deficiency. While this result is impressive in its generality, its implications are less clear.

One interpretation of this bias towards simple outputs, taking into account that GP maps can be described at many scales, is that simple patterns should be the rule in nature. As we have shown, simple maps produce simple outputs, and when both the inputs of a map and the map itself are the outputs of other simple maps, one quickly falls into a situation where *simplicity begets simplicity*. In a sense, these cumulative biases towards simplicity would seem to make complex patterns – and complex behaviour – unlikely, and arguably impossible.

Clearly, this interpretation seems to be at odds with reality: despite the biases towards simplicity, there are input-output maps in nature which produce chaotic and random outputs, and nature is full of simple systems which combine in order to produce larger-scale complex behaviour. GRNs are in fact a prime example of this *emergent complexity*, as they are behind cell signalling, differentiation, embryonic development, as well other complex biological patterns. This form of emergent behaviour does not seem to be described by the results presented in this thesis, and it is definitely a very promising direction to follow.

The quotation in the beginning of this chapter is a phrase by the French mathematician Marcel Schützenberger, who claimed there was a gap in the interpretation of evolution proposed by the modern synthesis. In his paper, Schützenberger contrasted the claim that biological evolution proceeds by random mutations to genotypes with our experience with computer software, which is notoriously susceptible to failure upon random mutations in the code. He argued that there was no biological principle which suggested that biological shape and function should be more robust to random mutations than computer software.

In 1966, when Schützenberger made these claims, Darwinian theory did not address GP maps, and often treated genotype and phenotype as the same object. The presence of correlations in GP maps, however, together with effects such as the arrival of the frequent, suggests that genetic robustness comes as it were for free in nature, i.e. as a by-product of the structural properties of GP maps. We believe the study of GP maps might address the gap in the theory of evolution as described by Schützenberger, and help explain how evolution works.

6.3 Evolution, machine learning, and computation

Throughout this thesis, some chapters were dedicated to GP maps, and others studied input-output maps more broadly, but we never made a clear distinction between which properties were unique to GP maps and which others were properties of a broader class of input-output maps. This is because we believe that the structural properties of GP maps presented in Chapter 1 are likely to be more general properties of input-output maps. More work needs to be done to test this hypothesis, but the diversity of GP maps that exhibit simllar structural properties is evidence in its favour.

One example of a parallel between a GP map and a (non-biological) input-output map is seen between GRNs and the matrix map as defined in Chapter 2. When GRNs are modelled using Boolean threshold networks, they are mathematically similar to the matrix map, which takes a binary vector as input, multiplies it by a matrix and then applies a threshold function to the result. This is also the building block of neural network models in machine learning [169]. This parallel between GRNs and artificial neural networks has been pointed out before [231, 218, 159]. In GRNs, the matrix entries represent the gene-gene interactions, while in neural networks they represent the neural network weights, but the conversion of inputs into outputs is the same in both cases: a linear transformation composed with a nonlinear threshold function.

An interesting aspect of these systems modelled by variations on the matrix map is that their outputs can also be fed into other matrix maps, gene networks or neural networks, which could result in an even stronger canalisation of many inputs into few outputs, and possibly stronger simplicity bias and randomness deficiency. In fact, this compositional property of matrix maps has already been observed in the context of GRNs, in works such

as the model of transcriptional gene networks proposed by Borenstein and Krakauer [32], who study the effect of building a multilayered matrix map on the resulting number of phenotypes. In the context of artificial neural networks, this compositionality is the base of *deep neural networks*.

This parallel between gene regulatory networks and artificial neural networks also implies something about evolution itself. Following the analogy between both kinds of networks, the evolutionary search for a gene network with a given set of biological properties can be compared to the machine learning task of finding a set of neural network weights that minimises a cost function. Random mutations on a genotype could be seen as random changes on the neural weight matrix, and the fitness "hill-climbing" behaviour could be compared to the gradient descent algorithms in machine learning [227].

The parallels between evolution and computation, in particular between evolution and (machine) learning, are discussed in a large body of literature [240], ranging from Leslie Valiant's work on formalising evolutionary dynamics under computational learning theory [227] to Daniel Dennett's view of natural selection as a substrate-neutral algorithm for moving through a design space [61]. We believe that the study of GP maps might contribute to this literature. Based on the work presented in this thesis, one could expect that phenomena such as the arrival of the frequent or simplicity bias might also be seen in supervised learning tasks, and that even more biologically relevant properties such as robustness and evolvability might have a computational or learning equivalent. Needless to say, the translation of concepts from one field to another leads to more questions than answers, and a lot of interesting work ahead – with neurons, bits, genes and machines.

Appendix A

Approximations to Kolmogorov complexity

In Chapter 2, we mentioned that Kolmogorov complexity is uncomputable, but also that several complexity measures based on lossless compression algorithms have been used to estimate it successfully [148]. In this section, we compare some of these measures, and show that the phenomenon of simplicity bias does not depend on the complexity estimator.

In Figure A.1a we plot the probability distribution of complexities for a wide range of string lengths, calculated with our Lempel-Ziv complexity approximator $C_{LZ}(x)$, defined in equation (2.17) in Chapter 2. We completely enumerated all strings of length $n \leq 30$, and for longer lengths we sampled 10^9 strings for each length. We estimated the mean and modal complexity, as well as its standard deviation, as a function of the length n. Also, we show in Figure A.1e that as the length of the strings get longer, the distributions of $C_{LZ}(x)$ get relatively narrower. The mean C_{LZ} is expected to approach $\langle C_{LZ} \rangle \approx n$ in the limit when $n \to \infty$ [252, 139]. The main thing to note is that for a given n, strings with complexity well below the mean are rare, and become rarer for lower complexities.

Regardless of the wide use of Lempel-Ziv complexity measures, there are some subtleties that should be kept in mind when interpreting results obtained with it. Firstly, rather than most strings having the maximum complexity measured by C_{LZ}, we find that the majority of strings are close to the mode, which is in turn very near the mean complexity (See Figure A.1c,d,e). For Lempel-Ziv it is known that strings with maximal complexity are somewhat anomalous because they can be created by an algorithmic process [75], which is an artefact of the Lempel-Ziv algorithm itself, and so are in fact not the most complex in a Kolmogorov sense. However, as seen in Figure A.1 these highest complexity strings remain rare, and so for our purposes they do not play a big role.

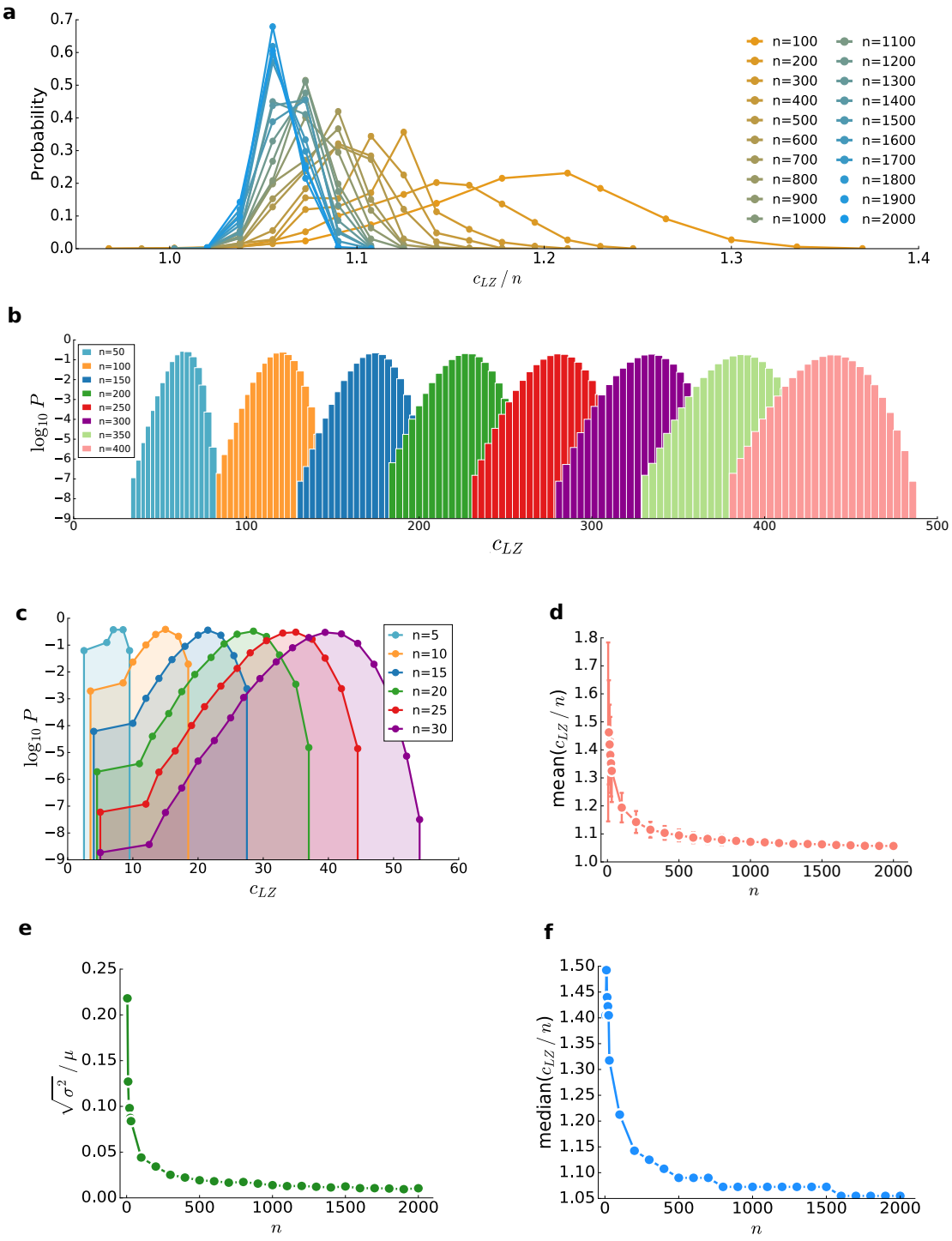


Figure A.1: **Distribution of complexity values calculated with $C_{LZ}(x)$ for strings of different length n.** All strings of length $n = \{5, 10, 15, 20, 25, 30\}$ were enumerated, and for $n > 30$ samples of 10^9 strings were taken. **(a)** Distribution of $C_{LZ}(x)/n$, for $n = 100$ to 2000. **(b)** Distribution of $C_{LZ}(x)$, for $n = 50$ to 400. **(c)** Distribution of $C_{LZ}(x)$, for $n = 5$ to 30. **(d)** Mean of $(C_{LZ}(x)/n)$, for $n = 5$ to 400. Error bars represent one standard deviation. **(e)** Standard deviation σ over the mean μ of $C_{LZ}(x)$, for $n = 5$ to 400. **(f)** Median of $(C_{LZ}(x)/n)$, for $n = 5$ to 400. Note that as n grows, the mean and median C_{LZ}/n approach 1, and the standard deviation σ over the mean μ drops: In other words, the distribution becomes more peaked. Lines connecting data points were added to guide the eye.

We also applied two alternate complexity measures to $C_{LZ}(x)$. Firstly, in Figure A.2 we apply the `Compress` function in *Mathematica* to RNA secondary structures of $n = 55$ and $n = 80$ bases. As can be seen, the values of the complexity approximation $\tilde{K}(x)$ are different, so the values of a and b estimated using equation (2.19) are different as well. Nevertheless, the same basic simplicity bias phenomenology is observed. Note that `Compress` is similar to `zlib` compression, which is based on another of Lempel and Ziv's famous compression algorithms, often called `LZ77` [251].

Given that most lossless compression algorithms are influenced by the ideas of Lempel and Ziv, it is not straightforward to find measures that are truly different in their core. Recently, however, a fundamentally different way of estimating the Kolmogorov complexity of strings has been derived in an important series of papers [60, 248, 212] that apply the full AIT coding theorem by sampling over many Turing machines. In principle this coding theorem method (CTM) is very powerful, and in particular can go well beyond lossless compression techniques, which are fundamentally sophisticated entropy measures. However, since the CTM relies on sampling, its accuracy is limited to very short strings ($\lesssim 12$ bits). To estimate the complexity for longer strings, the CTM is normally used in conjecture with the Block Decomposition Method (BDM), which estimates the complexity of a long string by breaking it into smaller strings, whose complexity can be calculated by the CTM [249, 212].

As shown in Figure A.2, the simplicity bias phenomenon is reproduced using these alternative complexity measures. For the BDM plot we simply fit both a and b, rather than using equation (2.19), as some of the simplifying assumptions used in Chapter 2 do not work for the BDM method. For example, we assume that a can be approximated by assuming $b = 0$, but that does not work here, most likely because there is a larger additive constant to the BDM complexity than to $C_{LZ}(x)$. However, as argued in Chapter 2, additive and multiplicative changes can all be absorbed into a and b, as long as they are $\mathcal{O}(1)$.

Finally, the coding theorem results which we invoke apply to the prefix-free version of Kolmogorov complexity, denoted $K(x)$, as opposed to the plain Kolmogorov complexity, denoted $C(x)$, see ref. [148]. However, while these two measures have important theoretical differences, they are asymptotically equal, as mentioned in Chapter 2. Since we approximate $K(x)$ anyway, we ignore the subtle distinction between these measures in our approximations. Some of the differences may be absorbed into our parameters a and b, as discussed previously.

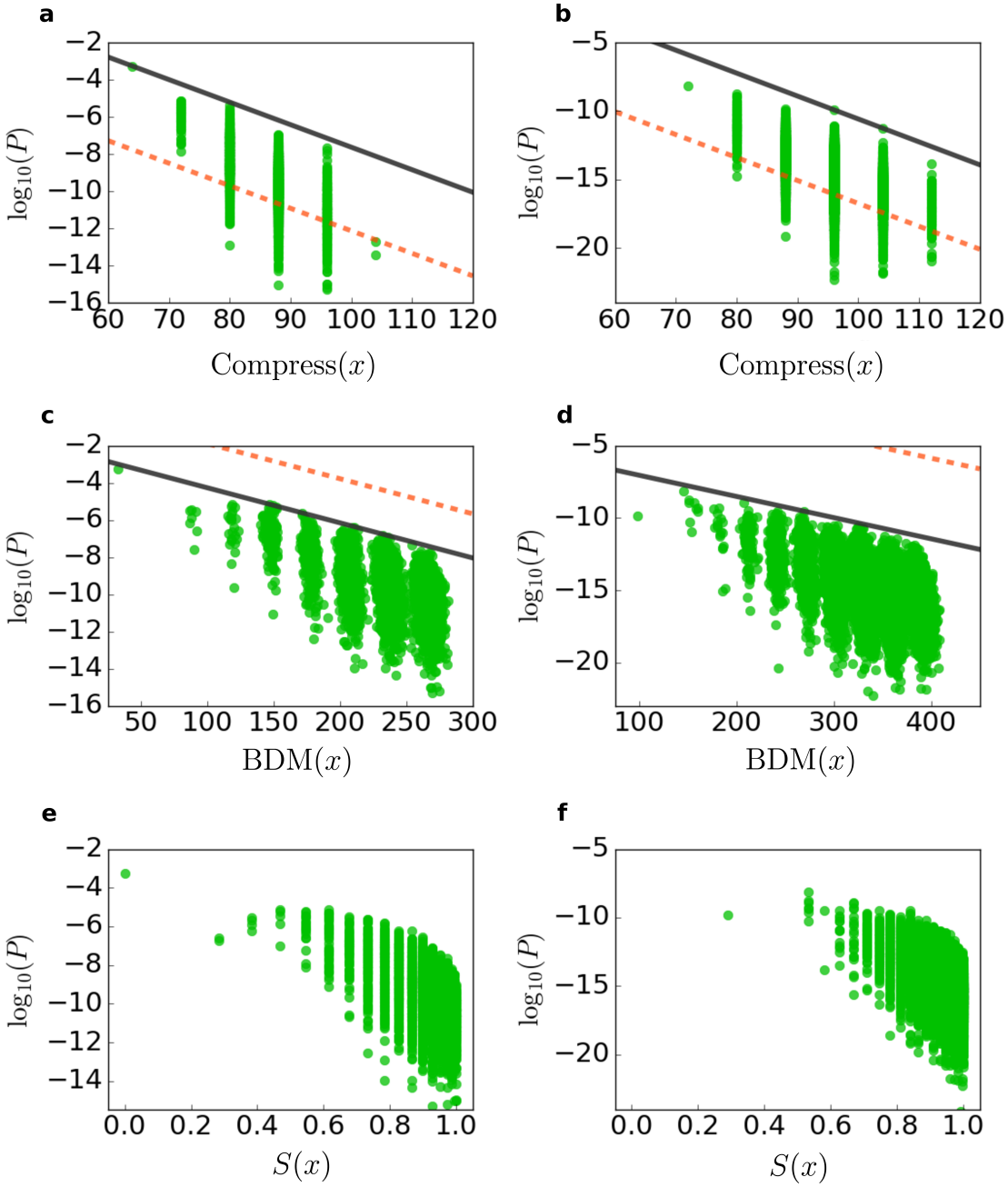


Figure A.2: **Simplicity bias predicted by other approximations to Kolmogorov complexity**. We use, *Mathematica*'s `Compress` function, the block decomposition method (BDM) and the simple entropy $S(x)$ of the dot-bracket notation, for $n = 55$ and $n = 80$ RNA secondary structures. In order, the plots show **(a)** `Compress` for $n = 55$ RNA, **(b)** `Compress` for $n = 80$ RNA, **(c)** BDM for $n = 55$ RNA, **(d)** BDM for $n = 80$ RNA, **(e)** Entropy $S(x)$ for $n = 55$ RNA, **(f)** Entropy $S(x)$ for $n = 80$ RNA. The solid lines denote our estimated upper bound, the dashed lines are the upper bound with $b = 0$. For the `Compress` and BDM method, we observe a similar simplicity bias phenomenology to what was observed in Figure 2.2, where C_{LZ} was used. For the entropy measure, there is also a decay of the probability with increasing complexity, but the behaviour of the upper part of the curves are significantly less linear than for `Compress`, BDM and $C_{LZ}(x)$.

Appendix B

Robustness of the NNSE results

Jörg et al's Neutral Network Size Estimator (NNSE) algorithm, described in Chapter 3, relies on a pair of parameters: `NTHERMAL`, the number of thermalization sweeps to be run before the nested Monte Carlo runs, and `NRUNS`, the number of Monte Carlo runs where the ratios $\frac{V(r)}{V(r+1)}$ are measured. In this appendix we confirm the robustness of this method to different choices `NTHERMAL` and `NRUNS`, for GRN Boolean threshold models with $N = 4$, $N = 5$ and $N = 6$ genes.

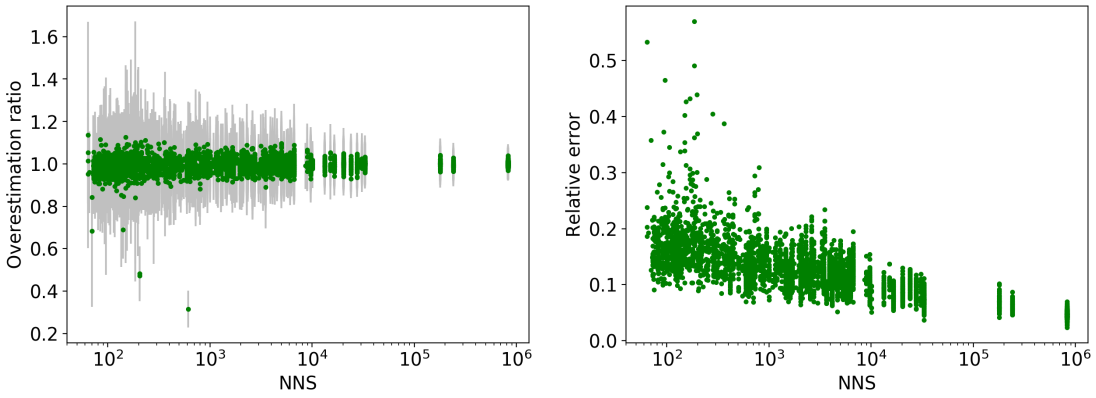


Figure B.1: Overestimation ratio **(left)** and relative error **(right)** of the NNSE plotted against neutral network sizes for various $N = 4$ GRN phenotypes, defined as the fine-grained BTN phenotypes in Chapter 3.

Since the NNSE is a sampling algorithm, it is expected that its error will be larger for smaller neutral networks. This difficulty in estimating lower neutral network sizes is confirmed in Figure B.1, which shows the ratio between estimated and measured NNS versus the measured NNS, as well as the relative error of the estimation, that is, the difference between the estimated and measured NNS, divided by the actual measured neutral network size. Both plots on Figure B.1 suggest that the overestimation and the error decrease as

neutral networks grow, with standard deviations rarely reaching 10% of the correct measured value.

It is also important to establish how the number of random walk iterations affect the convergence of the algorithm. To address that, we ran the algorithm for `NTHERMAL` and `NRUNS` going through values in $\{500, 1000, 2000, 5000, 10000, 20000, 50000\}$, in a total of $7 \times 7 = 49$ comparisons, ran for 20 phenotypes covering a range of neutral network sizes.

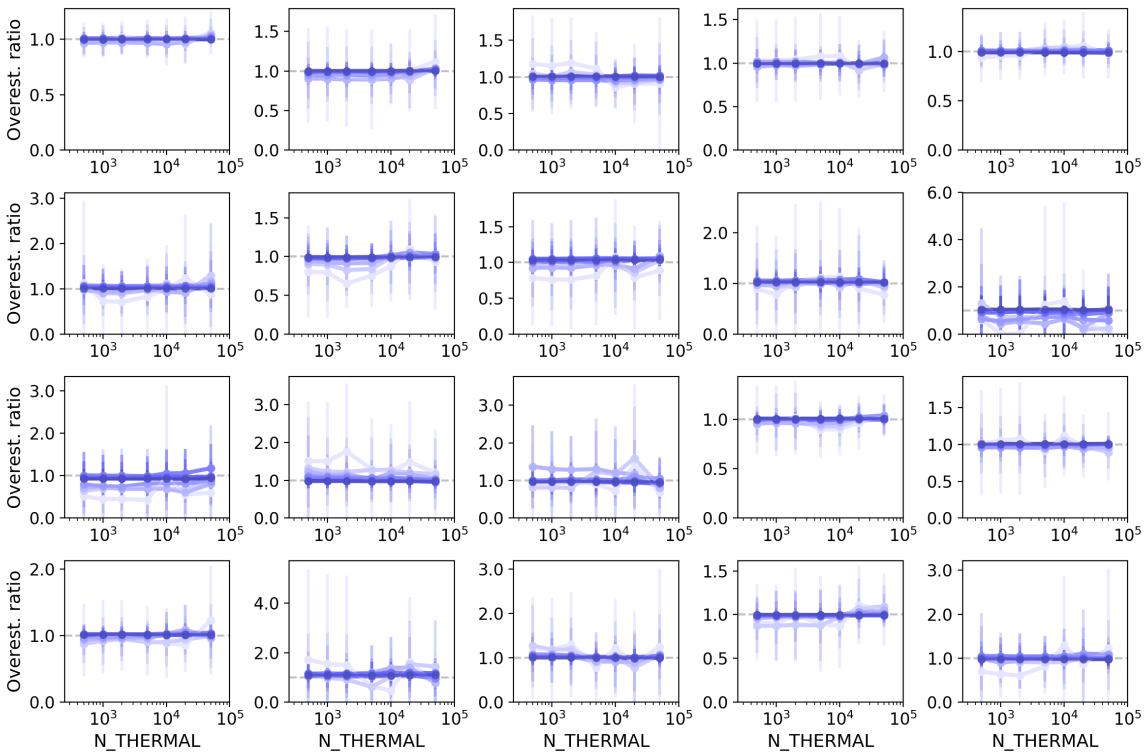


Figure B.2: `NTHERMAL` **does not seem to affect the overestimation ratio.** Each plot represents 49 runs of the NNSE for the same phenotype, with `NTHERMAL` and `NRUNS` varying from 500 to 50000. Darker curves indicate higher values of `NRUNS`. The average overestimation ratio remains around 1, regardless of `NTHERMAL`.

Figures B.2 and B.3 show the result of varying `NTHERMAL` and `NRUNS`. In both figures, each one of the 20 subplots represents 49 runs for the same phenotype, with the y axis representing the overestimation ratio of the phenotype's neutral network size, and the x axis representing `NTHERMAL`, in Figure B.2, and `NRUNS`, in Figure B.3, both ranging from 500 to 50000. Darker curves represent higher values for the parameter that is not represented in the x axis: `NRUNS` in Figure B.2, and `NTHERMAL` in B.3. The plots on both figures reveal the importance of each parameter: for most phenotypes, the overestimation ratio converges to a value of 1 (i.e., no overestimation) for `NRUNS` around 10^3 to 10^4 nested Monte Carlo

iterations, as seen in Figure B.3, while varying NTHERMAL does not seem to bear any effect on the overestimation ratio, drawing flat lines in Figure B.2.

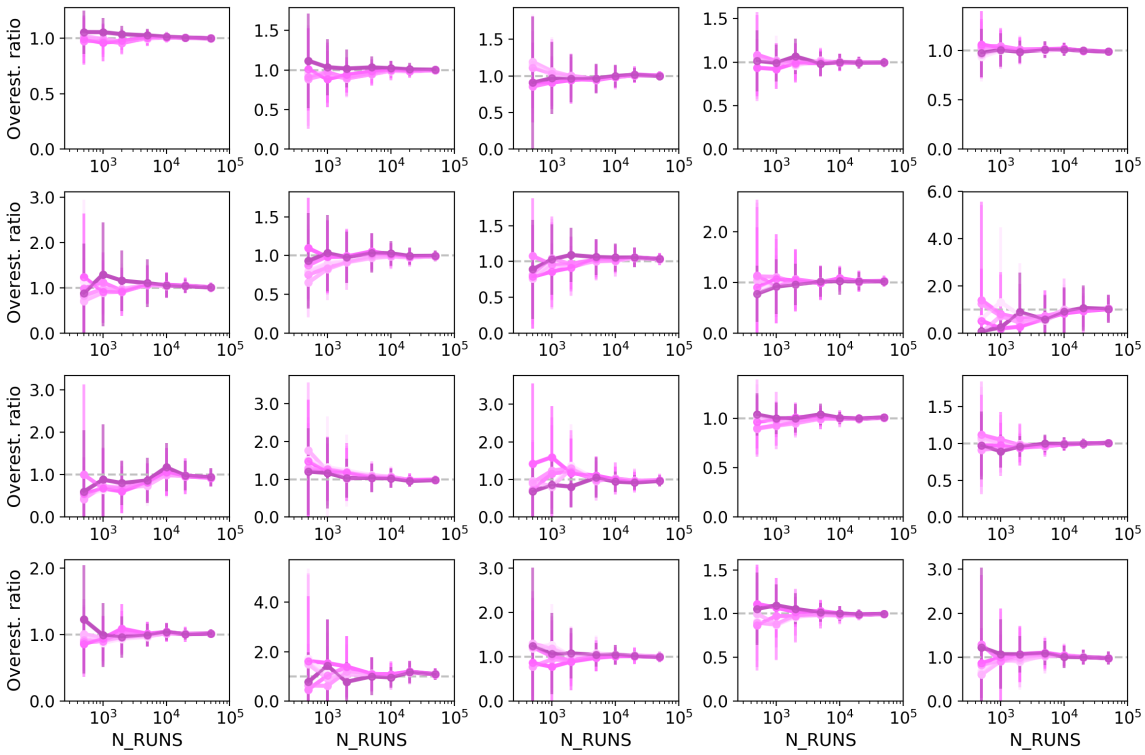


Figure B.3: **Higher NRUNS improves the overestimation ratio.** Each plot represents 49 runs of the NNSE for the same phenotype, with NTHERMAL and NRUNS varying from 500 to 50000. Darker curves indicate higher values of NTHERMAL. As the number of nested Monte Carlo iterations (NRUNS) increases to around 10^3-10^4, the overestimation ratio for most phenotypes converges to a value of 1 (i.e., no overestimation).

Another way to look at how the number of runs affects the NNSE estimation is the relative error between the estimated and measured NNS. From the law of large numbers, one would expect the relative error to be proportional to the square root of NRUNS. Figure B.4 shows exactly that, for the same 20 phenotypes mentioned above: the relative error plotted on the y axis decreases with a slope of approximately 0.50, proportionally to the square root of NRUNS.

The accuracy of the NNSE for $N = 4$ GRNs suggests this algorithm might perform well on larger genotype spaces. To produce a "ground truth" for the NNSE, we took a sample of 10^6 genotypes for $N = 5$ and $N = 6$ GRNs, from a total of $3^{25} \approx 8.5 \times 10^{11}$ genotypes and $3^{36} = 1.5^{17}$ genotypes respectively. Naturally, this sample is likely to overestimate the smaller neutral networks, as its estimate should be within one over the square root of the actual neutral network size, which is the same error resulting from the NNSE.

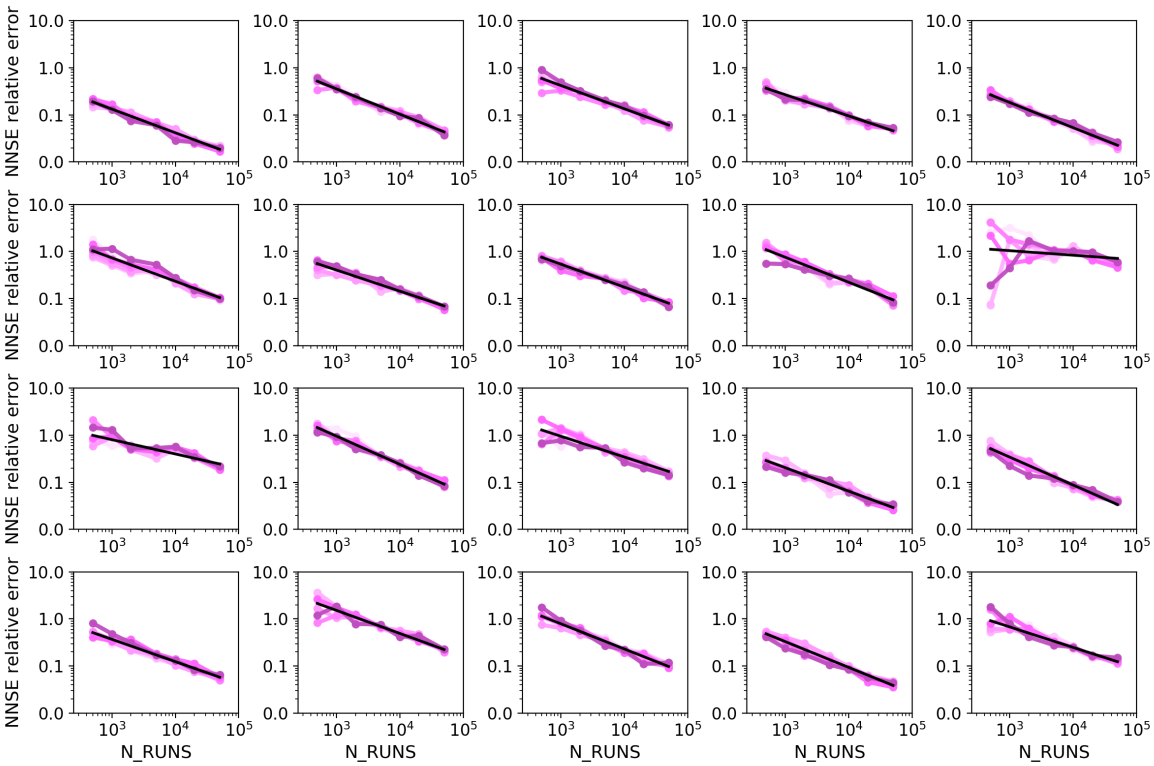


Figure B.4: The logarithm of the relative error of the NNSE prediction, plotted for the same 20 phenotypes presented in the figures above, falls with a slope of approximately 0.50 for most phenotypes, that is, proportionally to the square root of `NRUNS`. Darker curves indicate higher values of `NTHERMAL`, running from 500 to 50000.

Figure B.5 shows NNSE results for $N = 5$ and $N = 6$ GRNs respectively, for runs of 20 trials of 5×10^3 thermalisation steps, and 2×10^4 iterations. Both $N = 5$ and $N = 6$ show a reliable match between the NNSE estimate and the estimate obtained by randomly sampling genotypes. Just as in the results for $N = 4$, as the neutral network size increases, the overestimation ratio and relative error converge to 1 and 0 respectively. These results confirm the accuracy of this adaptation of the NNSE algorithm for larger genotype spaces.

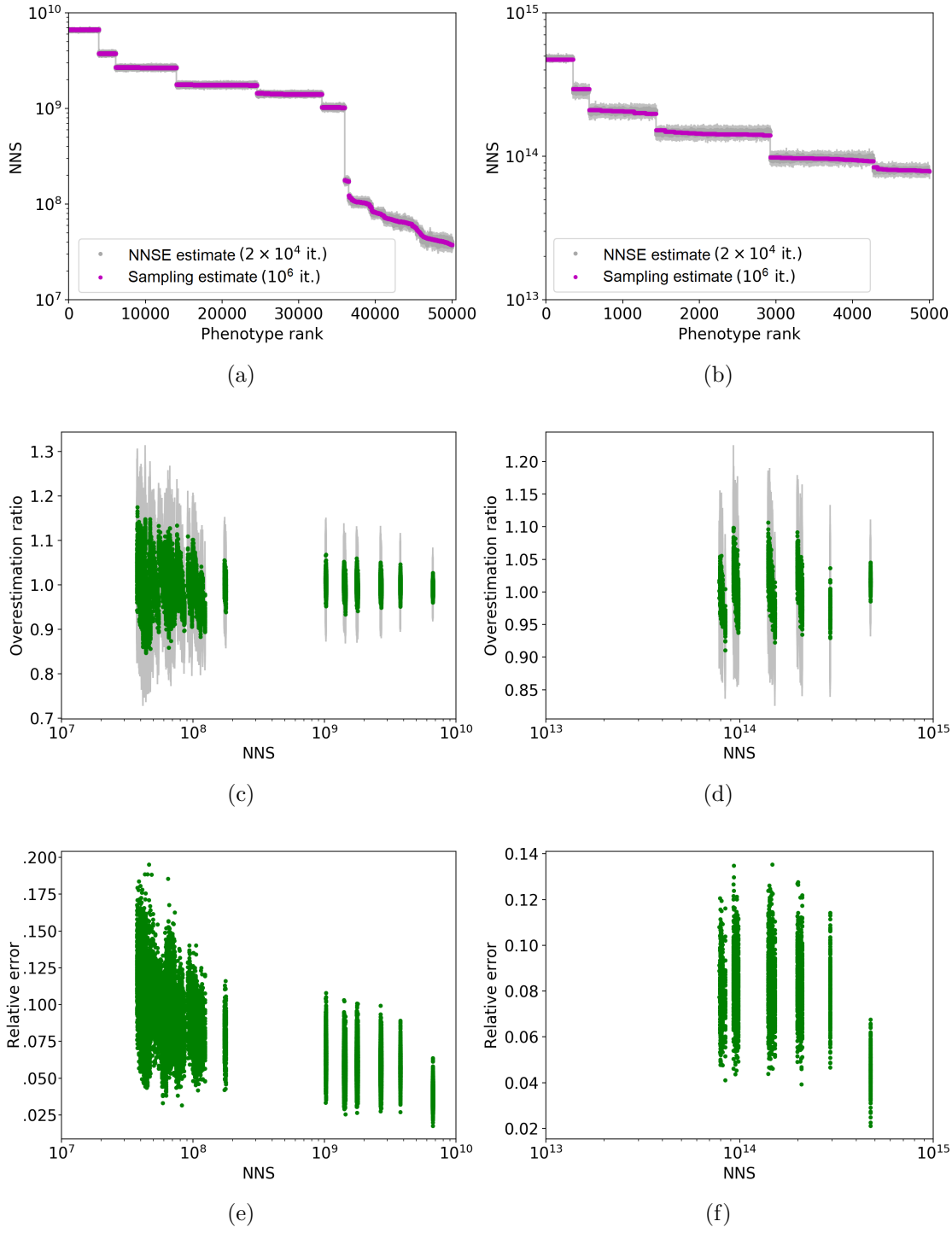


Figure B.5: **Comparison between the NNSE results against brute force sampling for $N = 5$ and $N = 6$ GRNs, shown in the left and right columns respectively.** The NNSE algorithm was run for 5×10^4 genotypes, over 2×10^4 iterations. **(a)** and **(b)**, also shown in Chapter 3, are rank plots comparing phenotype neutral network sizes (NNS) obtained by full enumeration of genotype space (shown in pink) to their results using the NNSE (shown in grey). **(c)** and **(d)** show the overestimation ratio of the NNSE for varying values of the actual neutral network size, and **(e)** and **(f)** show the relative error of the NNSE for varying neutral network sizes.

Appendix C

Robustness of the BIRCH results

In Chapter 4, we define GP maps for a series of GRNs using the BIRCH clustering algorithm [250]. In this appendix, we confirm that our results are robust to different choices of output representation and distance metric, as well as remaining qualitatively the same for different levels of output coarse-graining.

Trying different metrics

As discussed in section 4.2.3, the metric used to compare the outputs the ODE model is fundamental to any conclusion drawn from this study, as different metrics will better capture different features of the GP map. For example, if one were to compare outputs using Euclidean distance, concentration curves that are identical up to a phase shift will result in a large distance. For example, $\sin(t)$ and $\sin(t + \frac{\pi}{2})$ rarely coincide, even though both curves have the same functional form. In this case, any method based on a Fourier transform might yield more accurate results, in the sense of returning a small distance for pairs of outputs which biologically do not differ much.

Another method which addressses the problems with the Euclidean metric is the Dynamic Time Warping (DTW) method [22], which was developed to compare time series which might vary in speed by measuring the minimal amount of stretching or "warping" necessary to produce one curve from the other. The calculation of this "optimal warping" is a often implemented as a dynamic programming algorithm. For example, since the curves in Figure C.1 are out of phase with each other, Euclidean distance might return a high value. However, since the blue curve is a "slightly stretched" version of the red curve, DTW will return a low value.

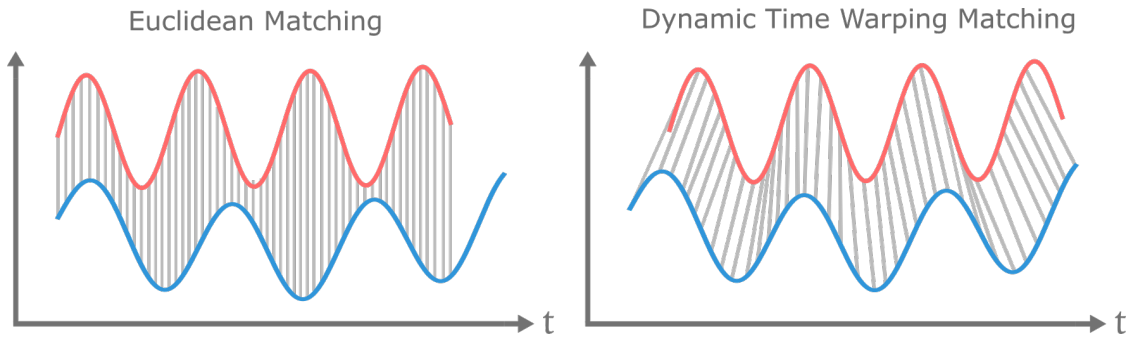


Figure C.1: Illustration of the difference between comparing time series using Euclidean distance and Dynamic Time Warping. While the former can only detect the difference between curves at a given time step, the latter "warps" one curve into another, "stretching" over the time axis.

The DTW method is considerably improved by the LB Keogh bound, which was introduced in 2002 by Eamonn Keogh [122] as a tool to estimate a lower bound for several time series distance measures, but for Dynamic Time Warping in particular. Due to the way it exploits dynamic programming constraints [200], it effectively speeds up DTW by orders of magnitude [150]. As a result, even though the DTW algorithm originally has a time complexity of $\mathcal{O}(n^2)$, LB Keogh effectively makes it DTW run in $\mathcal{O}(n)$ [210].

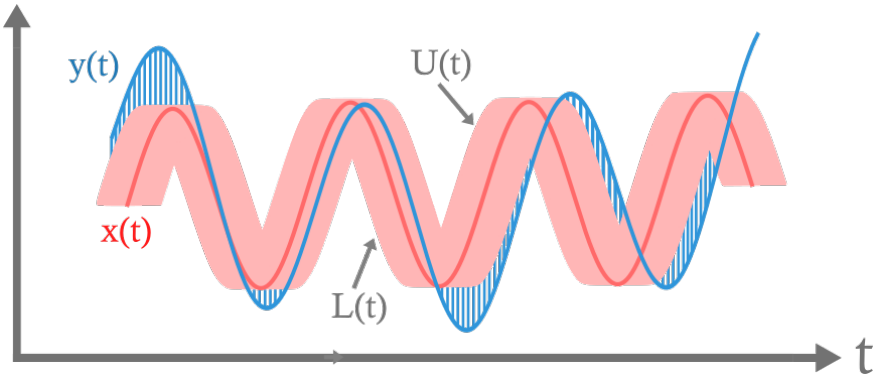


Figure C.2: Illustration of the LB Keogh method [122]: given two time series $x(t)$ and $y(t)$, the LB Keogh bound is defined as the Euclidean distance between $y(t)$ and the closest part of the envelope formed by the upper bound $U(t)$ and the lower bound $L(t)$ of $x(t)$.

Figure C.2 illustrates how LB Keogh works: given two time series $x(t)$ and $y(t)$, one draws two curves around $x(t)$, an upper bound $U(t)$ and a lower bound $L(t)$, defined as $U(t) = \max(\, x(t-r), \ldots, x(t+r)\,)$ and $L(t) = \min(\, x(t-r), \ldots, x(t+r)\,)$ for a window size r. This window was chosen here as $r = 5$ timesteps. Given the envelope formed by $U(t)$ and $L(t)$ around $x(t)$, the LB Keogh bound is given by the Euclidean distance between $y(t)$ and the closest part of the envelope, which is a (tight) lower bound to the DTW between

$x(t)$ and $y(t)$. The LB Keogh bound is also shown in equation (C.1).

$$\texttt{LB\_Keogh}(x,y) \;=\; \sum_{i=1}^{n} \begin{cases} (y(t_i) - U(t_i))^2 & \text{if } y(t_i) > U(t_i) \\ (y(t_i) - L(t_i))^2 & \text{if } y(t_i) < L(t_i) \\ 0 & \text{otherwise} \end{cases} \tag{C.1}$$

Although the DTW between two curves is a distance-like quantity measuring the "minimum warping" necessary to turn one curve into the other, it does not qualify as a distance, as it does not always satisfy the triangle inequality. We address this by using a symmetric version of LB Keogh, and calculate the distance between two time series $x(t)$ and $y(t)$ as the sum $\texttt{LB\_Keogh}(x,y) + \texttt{LB\_Keogh}(y,x)$.

In addition to the Euclidean and LB Keogh distances, we also compared time series using the SAX distance [151], standing for Symbolic Aggregate approXimation, which comes from discretising the range of values of the $x(t)$ and $y(t)$ into a series of intervals, effectively approximating both curves by piecewise constant functions, and then calculating a Hamming-like distance between both approximated curves.

Trying different output representations

Equally important as choosing the distance metric to be used by the clustering algorithm is choosing how to represent the outputs of the ODE model. A metric will only make sense when used with the right output representation. For instance, the most natural representation is to take the output curve $y(t)$ as produced by the model, as what we will here call the "raw curve". The problem involved with this representation is how it might lead to misleading results depending on the chosen distance metric, such as in the example of the functions $\sin(t)$ and $\sin(t + \frac{\pi}{2})$ above, where the Euclidean distance turned out to be a bad choice of metric. This choice of output representation and metric would also result in large distance values for curves which are just scaled versions of each other, such as $\sin(t)$ and $100\sin(t)$.

There are many alternatives to representing the ODE outputs as the raw curves: normalize all concentration curves so that they all remain bounded by $[0, 1]$, representing them in logarithmic scale, taking the Fourier transform of any of these (raw, normalised, logscale) curves, or even representing each time series by a piecewise linear interpolation [121]. A more thorough study would take other techniques from the vast literature on the representation, comparison and indexing of time series, which includes tools such as wavelet analysis

and singular value decomposition, as well as decompositions into other polynomial bases and other forms of representation.

We addressed these choices of metric and representation by measuring the distance between outputs using various combinations of output representation and distance metric, represented in Table C. Note that the table does not include all possible combinations of the forms of data representation and distance metric listed in its columns. This is for two reasons. First, because the LB Keogh measure does not yield meaningful results when comparing anything other than time series. Second, because some combinations of data and metric result in too much coarse-graining of the data, which gives these metrics less discriminatory power than simply comparing the raw data using Euclidean distance. An example of one of these combinations that resulted in weaker distance measures combinations is using SAX distance to compare the raw data in log-scale. These "weaker" combinations of representation and metric were thus omitted.

Finally, since there is more than one way to define the biologically meaningful features of a distance metric, rather than trying to find the ideal combination of metric and output representation, it is better to look at which combinations produce similar results across different ODE systems, as those are likely to result in the same clustering of phenotypes. We performed this comparison taking 200 output curves for two GRNs, namely Lee et al's model for signalling in the Wnt pathway in Xenopus [136] and Leloup et al's model for the *Drosophila* circadian rhythm [137].

Figure C.3 shows the complete comparison for Lee et al's model. The top left of the figure shows a high correlation between the distance values that result from the first seven combinations of representation and metric, all presented in Table C. These combinations include the raw output curves and Euclidean data, and the raw outputs and the symmetric LB Keogh distance, but do not include the piecewise linear interpolation of the normalised outputs or the SAX metric. Leloup et al's model shows the same relation between metrics and output representations, where the first seven combinations of metric and representation all correlate with one another, whereas the last four alternatives do not seem to correlate with any other combination.

These results might seem like they indicate a fragile dependence on the exact measures chosen, but in fact they shine a light on what makes the output curves different from each other. This is because the piecewise linear interpolation (PLI) and the SAX metric, present

	Data	Distance metric
1	raw outputs	Symmetric LB Keogh
2	raw outputs	Euclidean distance
3	FFT (raw outputs)	Euclidean distance
4	PLI (raw outputs)	Euclidean distance
5	normalised outputs	Symmetric LB Keogh
6	normalised outputs	Euclidean distance
7	FFT (normalised outputs)	Euclidean distance
8	PLI (normalised outputs)	Euclidean distance
9	PLI (normalised outputs) in log-scale	Euclidean distance
10	raw outputs	SAX distance
11	normalised outputs	SAX distance

Table C.1: All the combinations of output representations and distance metrics used to cluster ODE outputs. For the normalised outputs, each curve is normalised by its maximum value. FFT and PLI stand respectively for fast Fourier Transform and piecewise linear interpolation [121].

in the four combinations that do not correlate with the rest of the results, are notably more coarse-graining than the other representations and metrics present in Table C, and therefore cannot capture the same differences between time series as the Euclidean or Symmetric LB Keogh distances can. Once this difference in coarse-graining is taken into consideration, the fact that the distance measurements involving PLI and SAX do not correlate with the ones in the green shaded area in Figure C.3 suggests that there are differences between the ODE output curves which would go unnoticed if one were to use PLI or SAX to calculate the distance between outputs.

The inefficacy of PLI and SAX is an important result for this analysis, as it speaks about the data we are comparing. It suggests that the differences between ODE outputs can be measured by Euclidean distance just as well as by the LB Keogh distance, but not as well by methods which coarse-grain the data such as PLI or SAX. With that in mind, the remaining analysis of this GP map was performed using the raw curves and Euclidean distance.

Trying different levels of coarse-graining

Similar to what we showed for the binary string phenotype in the previous section, varying the level of coarse-graining of the output time series also does not affect the results. Figure C.4 illustrates the results of varying the discretisation of the outputs for two of the

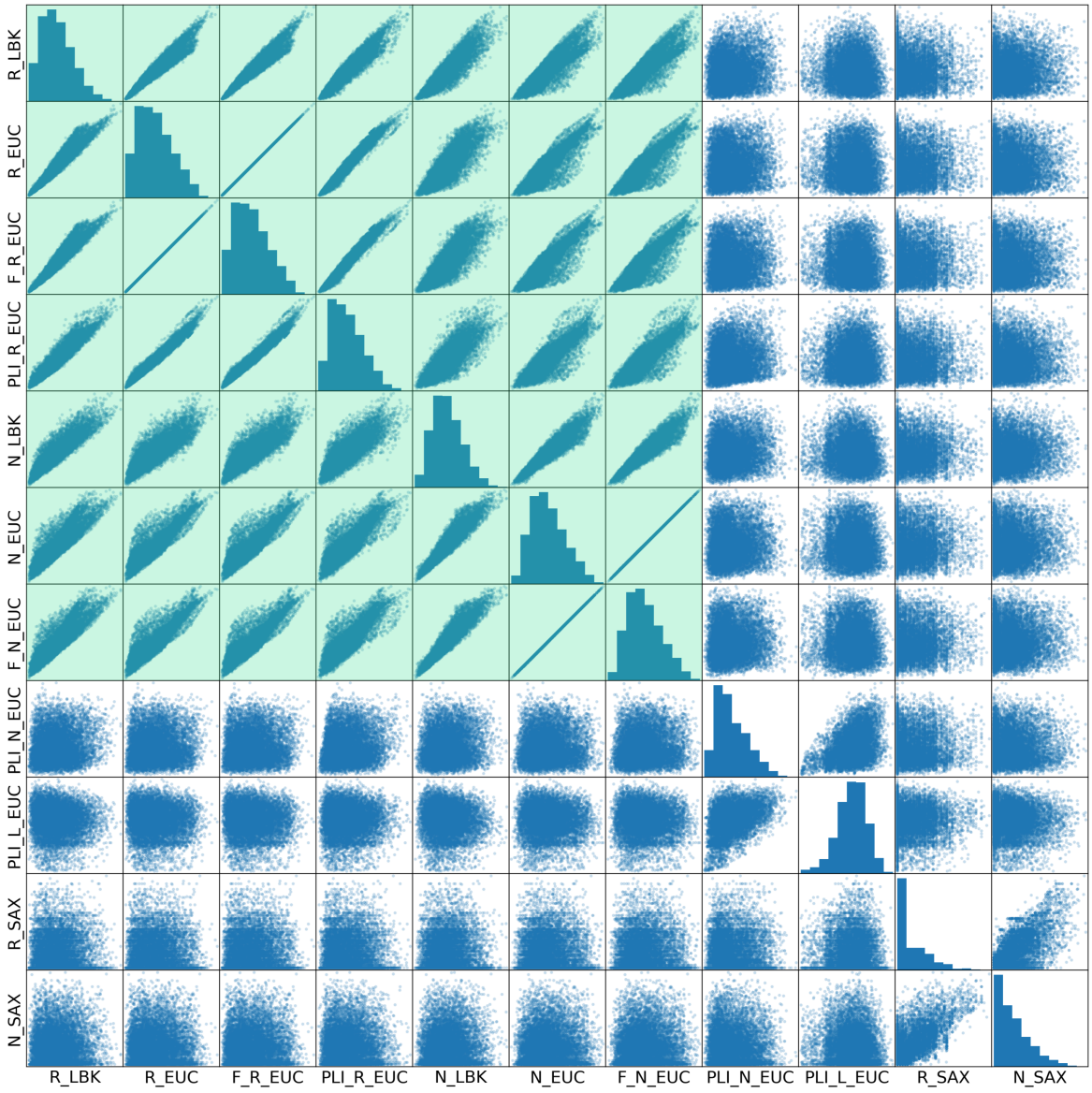


Figure C.3: **Comparing distance measurements different choices of metric and representation**. The green shaded area shows pairs of distance metric and output representation that show a high correlation, suggesting that the choice of metric and output representation dows not affect our results, as long as it stays within the combinations in the green shaded area. Distance measurements were calculated for 200 output curves for Lee et al's GRN model [136]. All metrics and output representations are listed in Table C, and reproduced as follows. `R_LBK`: raw outputs with LB Keogh distance, `R_EUC`: raw outputs with Euclidean distance, `F_R_EUC`: Fourier transform of the raw outputs with Euclidean distance, `PLI_R_EUC`: piecewise linear interpolation of the raw outputs with Euclidean distance, `N_LBK`: normalised outputs with LB Keogh distance, `N_EUC`: normalised outputs with Euclidean distance, `F_N_EUC`: Fourier transform of the normalised outputs, `PLI_N_EUC`: piecewise linear interpolation of the normalised outputs with Euclidean distance, `PLI_L_EUC`: piecewise linear interpolation of the normalised outputs in logscale with Euclidean distance, `R_SAX`: raw outputs with SAX distance, `N_SAX`: normalised outputs with SAX distance.

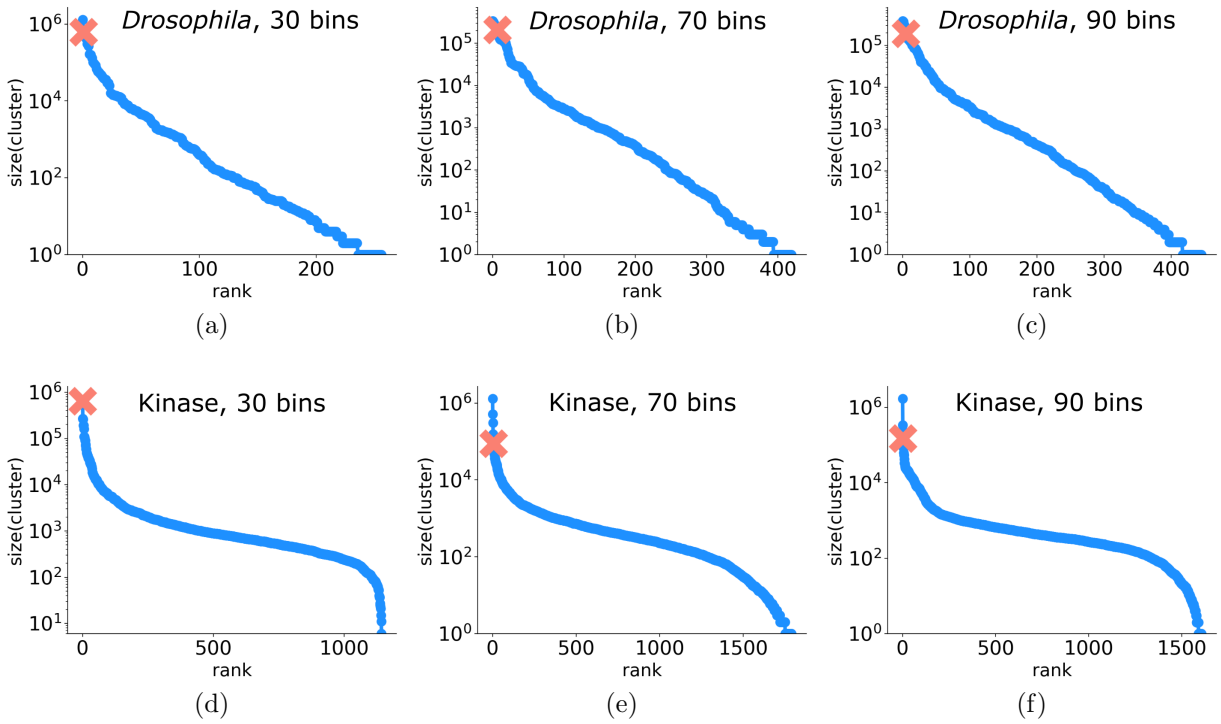


Figure C.4: **Distributions of phenotype cluster size for different output discretisations.** The plots show the GRNs in Ueda et al. [226] (top row) and Kholodenko [123] (bottom row), for varying levels of discretisation of the outputs (30, 70 and 90 bins, from left to right), showing that our results are not affected by the level of coarse-graining of the outputs.

ODE models, representing the *Drosophila* circadian rhythm [226] and a protein kinase cascade [123], with the number of time steps in the discretised time series from 30 to 70 and 90 time steps. The increasing fine-graining of the time series does not have a significant effect on the distribution of neutral network sizes.

Trying different parameters for BIRCH

We also ensured our results hold regardless of the parameters of the BIRCH algorithm, namely its cluster threshold and branching factor. Figure C.5 shows the neutral network size rank distribution for the GRNs in refs. [42], [226] and [123], for cluster thresholds ranging from 0.2 to 0.8 and cluster branching factors ranging from 10 to 50. Naturally, these parameters do affect the rank distributions, but they do not change the fact that neutral network sizes are distributed over orders of magnitude, nor the fact that the wild-type phenotype has one of the largest neutral networks, for the three GRNs presented in Figure C.5.

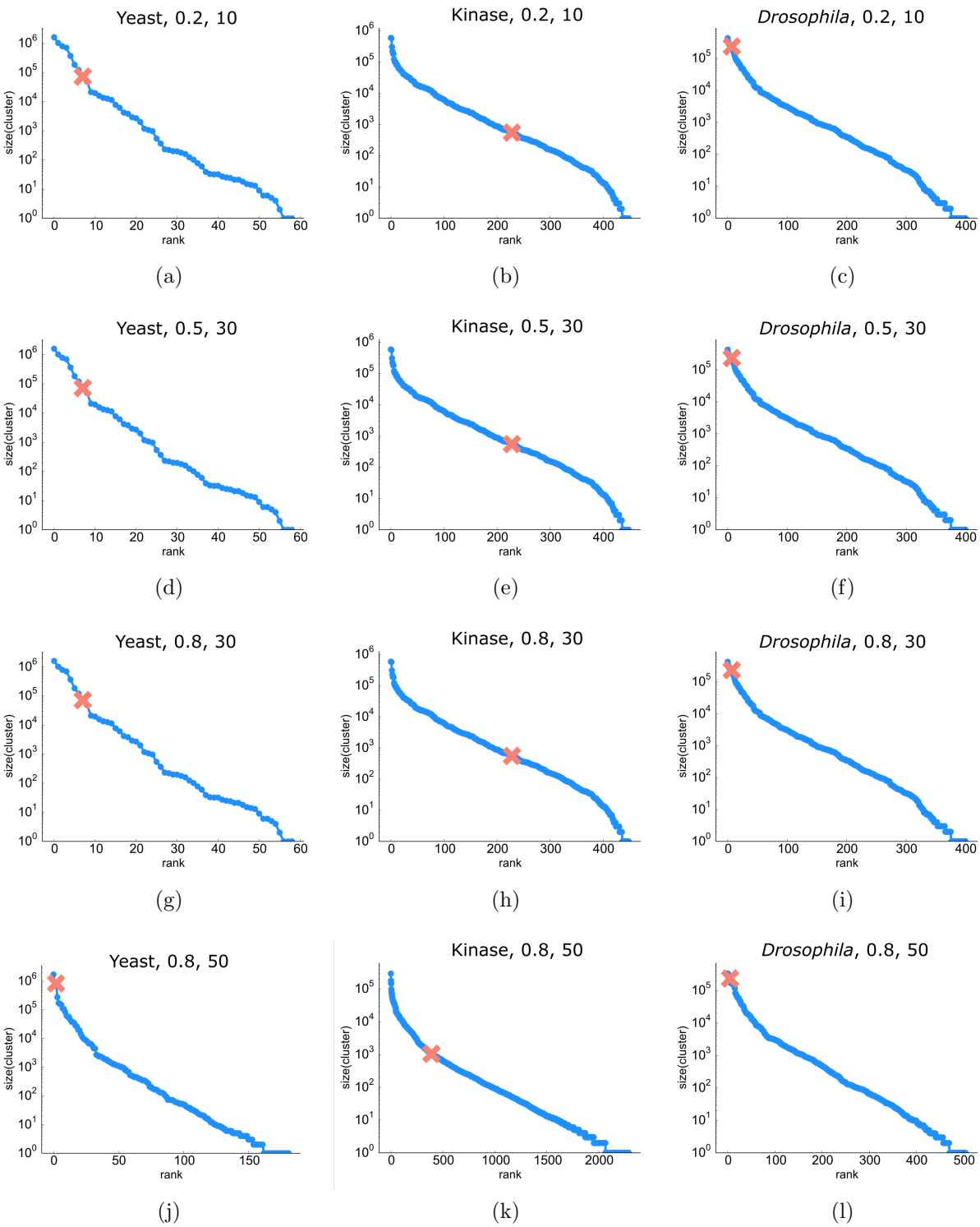


Figure C.5: **Distributions of phenotype cluster size for different parameters of the BIRCH model.** The plots show the GRN models in the work of Chen et al. [42] (left column), Kholodenko [123] (centre column) and Ueda et al. [226] (right column), for varying cluster thresholds and branching factors. From first to last row, the cluster threshold varies as $(0.2, 0.5, 0.8, 0.8)$ and the cluster branching factor varies as $(10, 30, 30, 50)$.

Bibliography

[1] Christoph Adami. Information theory in molecular biology. *Physics of Life Reviews*, 1(1):3–22, 2004.

[2] Christoph Adami. The use of information theory in evolutionary biology. *Annals of the New York Academy of Sciences*, 1256(1):49–65, 2012.

[3] Cristoph Adami and Nicolas J Cerf. Physical complexity of symbolic sequences. *Physica D*, 137(1):62–69, 2000.

[4] Jacobo Aguirre, Javier M Buldú, and Susanna C Manrubia. Evolutionary dynamics on networks of selectively neutral genotypes: Effects of topology and sequence stability. *Physical Review E*, 80(6):066112, 2009.

[5] Jacobo Aguirre, Javier M Buldú, Michael Stich, and Susanna C Manrubia. Topological structure of the space of phenotypes: the case of RNA neutral networks. *PLOS ONE*, 6(10):e26324, 2011.

[6] Sebastian E Ahnert. Structural properties of genotype–phenotype maps. *Journal of The Royal Society Interface*, 14(132):20170275, 2017.

[7] Sebastian E Ahnert, Iain G Johnston, Thomas MA Fink, Jonathan PK Doye, and Ard A Louis. Self-assembly, modularity, and physical complexity. *Physical Review E*, 82(2):026117, 2010.

[8] Pere Alberch. From genes to phenotype: dynamical systems and evolvability. *Genetica*, 84(1):5–11, 1991.

[9] Réka Albert and Hans G Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster. *Journal of theoretical biology*, 223(1):1–18, 2003.

[10] Maximino Aldana and Philippe Cluzel. A natural class of robust networks. *Proceedings of the National Academy of Sciences*, 100(15):8710–8714, 2003.

[11] Maximino Aldana, Susan Coppersmith, and Leo P Kadanoff. Boolean dynamics with

random couplings. In *Perspectives and Problems in Nolinear Science*, pages 23–89. Springer, 2003.

[12] Maximino Aldana and Hernán Larralde. Phase transitions in scale-free neural networks: Departure from the standard mean-field universality class. *Physical Review E*, 70(6):066130, 2004.

[13] José M Amigó, Janusz Szczepański, Elek Wajnryb, and Maria V Sanchez-Vives. Estimating the entropy rate of spike trains via lempel-ziv complexity. *Neural Computation*, 16(4):717–736, 2004.

[14] Neşe Aral and Alkan Kabakçıoğlu. Coherent regulation in yeast's cell-cycle network. *Physical biology*, 12(3):036002, 2015.

[15] Neşe Aral and Alkan Kabakçıoğlu. Coherent organization in gene regulation: a study on six networks. *Physical biology*, 13(2):026006, 2016.

[16] Ricardo BR Azevedo, Rolf Lohaus, Suraj Srinivasan, Kristen K Dang, and Christina L Burch. Sexual reproduction selects for robustness and negative epistasis in artificial gene networks. *Nature*, 440(7080):87, 2006.

[17] Enrique Balleza, Elena R Alvarez-Buylla, Alvaro Chaos, Stuart Kauffman, Ilya Shmulevich, and Maximino Aldana. Critical dynamics in genetic regulatory networks: examples from four kingdoms. *PLOS ONE*, 3(6):e2456, 2008.

[18] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews. Genetics*, 5(2):101, 2004.

[19] Ugo Bastolla and Giorgio Parisi. Relevant elements, magnetization and dynamical properties in Kauffman networks: A numerical study. *Physica D: Nonlinear Phenomena*, 115(3):203–218, 1998.

[20] Otto G Berg and Peter H von Hippel. Selection of DNA binding sites by regulatory proteins. *Trends in biochemical sciences*, 13(6):207–211, 1988.

[21] Aviv Bergman and Mark L Siegal. Evolutionary capacitance as a general feature of complex gene networks. *Nature*, 424(6948):549, 2003.

[22] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[23] Ashish Bhan, David J Galas, and T Gregory Dewey. A duplication growth model of gene expression networks. *Bioinformatics*, 18(11):1486–1493, 2002.

[24] William Bialek. *Biophysics: searching for principles*. Princeton University Press, 2012.

[25] William Bialek, Andrea Cavagna, Irene Giardina, Thierry Mora, Edmondo Silvestri, Massimiliano Viale, and Aleksandra M Walczak. Statistical mechanics for natural flocks of birds. *Proceedings of the National Academy of Sciences*, 109(13):4786–4791, 2012.

[26] JM Bibby, JT Kent, and KV Mardia. Multivariate analysis, 1979.

[27] Christine P Bird, Barbara E Stranger, Maureen Liu, Daryl J Thomas, Catherine E Ingle, Claude Beazley, Webb Miller, Matthew E Hurles, and Emmanouil T Dermitzakis. Fast-evolving noncoding sequences in the human genome. *Genome biology*, 8(6):R118, 2007.

[28] Tomas Björk. *Arbitrage theory in continuous time*. Oxford University Press, 2009.

[29] Gunnar Boldhaus, Nils Bertschinger, Johannes Rauh, Eckehard Olbrich, and Konstantin Klemm. Robustness of Boolean dynamics under knockouts. *Physical Review E*, 82(2):021916, 2010.

[30] Gunnar Boldhaus and Konstantin Klemm. Regulatory networks and connected components of the neutral space. *The European Physical Journal B-Condensed Matter and Complex Systems*, 77(2):233–237, 2010.

[31] Russell Bonduriansky and Troy Day. Nongenetic inheritance and its evolutionary implications. *Annual Review of Ecology, Evolution, and Systematics*, 40, 2009.

[32] Elhanan Borenstein and David C Krakauer. An end to endless forms: epistasis, phenotype distribution bias, and nonuniform evolution. *PLoS computational biology*, 4(10):e1000202, 2008.

[33] Peter J Bowler. *Evolution: the history of an idea*. University of California Press, 1989.

[34] Kevin S Brown, Colin C Hill, Guillermo A Calero, Christopher R Myers, Kelvin H Lee, James P Sethna, and Richard A Cerione. The statistical mechanics of complex signaling networks: nerve growth factor signaling. *Physical biology*, 1(3):184, 2004.

[35] Kevin S Brown and James P Sethna. Statistical mechanical approaches to models with many poorly known parameters. *Physical Review E*, 68(2):021904, 2003.

[36] J Burns. The synthetic problem and the genotype-phenotype relation in cellular metabolism. *Organization Stability and Process*, 3:47, 1970.

[37] Cristian S Calude. *Information and randomness: an algorithmic perspective*. Springer Science & Business Media, 1994.

[38] Ricky Chachra, Mark K Transtrum, and James P Sethna. Structural susceptibility and separation of time scales in the Van der Pol oscillator. *Physical Review E*, 86(2):026712, 2012.

[39] Gregory. J. Chaitin. *Algorithmic Information Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1987.

[40] Christophe Chassagnole, Naruemol Noisommit-Rizzi, Joachim W Schmid, Klaus Mauch, and Matthias Reuss. Dynamic modeling of the central carbon metabolism of Escherichia coli. *Biotechnology and bioengineering*, 79(1):53–73, 2002.

[41] Hao Chen, Guanyu Wang, Rahul Simha, Chenghang Du, and Chen Zeng. Boolean models of biological processes explain cascade-like behavior. *Scientific reports*, 7, 2016.

[42] Katherine C Chen, Laurence Calzone, Attila Csikasz-Nagy, Frederick R Cross, Bela Novak, and John J Tyson. Integrative analysis of cell cycle control in budding yeast. *Molecular biology of the cell*, 15(8):3841–3862, 2004.

[43] Xin Chen, Brent Francia, Ming Li, Brian Mckinnon, and Amit Seker. Shared information and program plagiarism detection. *IEEE Transactions on Information Theory*, 50(7):1545–1551, 2004.

[44] Dominique Chu, Nicolae Radu Zabet, and Boris Mitavskiy. Models of transcription factor binding: sensitivity of activation functions to model assumptions. *Journal of Theoretical Biology*, 257(3):419–429, 2009.

[45] Alonzo Church. A note on the entscheidungsproblem. *The journal of symbolic logic*, 1(1):40–41, 1936.

[46] Alonzo Church. An unsolvable problem of elementary number theory. *American journal of mathematics*, 58(2):345–363, 1936.

[47] Stefano Ciliberti, Olivier C Martin, and Andreas Wagner. Innovation and robustness in complex regulatory gene networks. *Proceedings of the National Academy of Sciences*, 104(34):13591–13596, 2007.

[48] Stefano Ciliberti, Olivier C Martin, and Andreas Wagner. Robustness can evolve gradually in complex regulatory gene networks with varying topology. *PLoS computational biology*, 3(2):e15, 2007.

[49] Rudi Cilibrasi, Paul Vitányi, and Ronald De Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.

[50] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Resilience of the internet to random breakdowns. *Physical review letters*, 85(21):4626, 2000.

[51] Gavin C Conant and Andreas Wagner. Convergent evolution of gene circuits. *Nature genetics*, 34(3):264, 2003.

[52] Otto X Cordero and Paulien Hogeweg. Feed-forward loop circuits as a side effect of genome evolution. *Molecular biology and evolution*, 23(10):1931–1936, 2006.

[53] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[54] Matthew C Cowperthwaite, Evan P Economo, William R Harcombe, Eric L Miller, and Lauren Ancel Meyers. The ascent of the abundant: how mutational networks constrain evolution. *PLoS computational biology*, 4(7):e1000110, 2008.

[55] Toby S Cubitt, David Perez-Garcia, and Michael M Wolf. Undecidability of the spectral gap. *Nature*, 528(7581):207–211, 2015.

[56] Bryan C Daniels, Yan-Jiun Chen, James P Sethna, Ryan N Gutenkunst, and Christopher R Myers. Sloppiness, robustness, and evolvability in systems biology. *Current opinion in biotechnology*, 19(4):389–395, 2008.

[57] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or Preservation of Favoured Races in the Struggle for Life*. Murray, 1859.

[58] Maria I Davidich and Stefan Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLOS ONE*, 3(2):e1672, 2008.

[59] Maria I Davidich and Stefan Bornholdt. Boolean network model predicts knockout mutant phenotypes of fission yeast. *PLOS ONE*, 8(9):e71786, 2013.

[60] J.P. Delahaye and H. Zenil. Numerical evaluation of algorithmic complexity for short strings: A glance into the innermost structure of algorithmic randomness. *Appl. Math. Comput.*, 219:63–77, 2012.

[61] Daniel C Dennett. Darwin's dangerous idea. *Recherche*, 27(293):100, 1996.

[62] B Derrida and H Flyvbjerg. The random map model: a disordered model with deterministic dynamics. *Journal de Physique*, 48(6):971–978, 1987.

[63] B Derrida and G Weisbuch. Evolution of overlaps between configurations in random Boolean networks. *Journal de physique*, 47(8):1297–1303, 1986.

[64] Bernard Derrida and Yves Pomeau. Random networks of automata: a simple annealed approximation. *EPL (Europhysics Letters)*, 1(2):45, 1986.

[65] L Peter Deutsch. DEFLATE compressed data format specification version 1.3. 1996.

[66] Kamaludin Dingle. *Probabilistic bias in genotype-phenotype maps*. PhD thesis, University of Oxford, 2014.

[67] Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. Input–output maps are strongly biased towards simple outputs. *Nature Communications*, 9(1):761, 2018.

[68] Kamaludin Dingle, Steffen Schaper, and Ard A Louis. The structure of the genotype–phenotype map strongly constrains the evolution of non-coding RNA. *Interface focus*, 5(6):20150053, 2015.

[69] Jeremy Draghi and Gunter P Wagner. The evolutionary dynamics of evolvability in a gene network model. *Journal of evolutionary biology*, 22(3):599–611, 2009.

[70] Genevieve Dupont and Albert Goldbeter. Modelling oscillations and waves of cytosolic calcium. *Nonlinear Analysis: Theory, Methods & Applications*, 30(3):1781–1792, 1997.

[71] David J Earl and Michael W Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

[72] Stuart J Edelstein, Olivier Schaad, Eric Henry, Daniel Bertrand, and Jean-Pierre Changeux. A kinetic mechanism for nicotinic acetylcholine receptors based on multiple allosteric transitions. *Biological cybernetics*, 75(5):361–379, 1996.

[73] Jeremy L England. Statistical physics of self-replication. *The Journal of chemical physics*, 139(12):09B623_1, 2013.

[74] Carlos Espinosa-Soto, Pablo Padilla-Longoria, and Elena R Alvarez-Buylla. A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. *The Plant Cell*, 16(11):2923–2939, 2004.

[75] E Estevez-Rams, R Lora Serrano, B Aragón Fernández, and I Brito Reyes. On the non-randomness of maximum Lempel Ziv complexity sequences of finite size. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 23(2):023118, 2013.

[76] Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–e131, 2006.

[77] David Fell and Athel Cornish-Bowden. *Understanding the control of metabolism*, volume 2. Portland press London, 1997.

[78] Evandro Ferrada and Andreas Wagner. A comparison of genotype-phenotype maps for RNA and proteins. *Biophysical journal*, 102(8):1916–1925, 2012.

[79] Paolo Ferragina, Raffaele Giancarlo, Valentina Greco, Giovanni Manzini, and Gabriel Valiente. Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment. *BMC bioinformatics*, 8(1):252, 2007.

[80] Thomas MA Fink, Karen Willbrand, and Francis CS Brown. 1-d random landscapes and non-random data series. *EPL (Europhysics Letters)*, 79(3):38006, 2007.

[81] Ronald Aylmer Fisher. *The genetical theory of natural selection: a complete variorum edition*. Oxford University Press, 1930.

[82] Henrik Flyvbjerg and NJ Kjr. Exact solution of Kauffman's model with connectivity one. *Journal of Physics A: Mathematical and General*, 21(7):1695, 1988.

[83] Markus Friedrich. Evo-devo gene toolkit update: at least seven Pax transcription factor subfamilies in the last common ancestor of bilaterian animals. *Evolution & development*, 17(5):255–257, 2015.

[84] Martin Fussenegger, James E Bailey, and Jeffrey Varner. A mathematical model of caspase function in apoptosis. *Nature biotechnology*, 18(7):768–774, 2000.

[85] Peter Gács. *Lecture notes on descriptional complexity and randomness*. Boston University, Graduate School of Arts and Sciences, Computer Science Department, 1988.

[86] Carlos Gershenson. Introduction to random Boolean networks. *arXiv preprint nlin/0408006*, 2004.

[87] Clare E Giacomantonio and Geoffrey J Goodhill. A Boolean model of the gene regulatory network underlying mammalian cortical area development. *PLoS computational biology*, 6(9):e1000936, 2010.

[88] Scott F Gilbert. The morphogenesis of evolutionary developmental biology. *International Journal of Developmental Biology*, 47(7-8):467, 2003.

[89] Grace Gill. Regulation of the initiation of eukaryotic transcription. *Essays in biochemistry*, 37:33–43, 2001.

[90] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434, 1976.

[91] Leon Glass and Michael C Mackey. *From clocks to chaos: the rhythms of life*. Princeton University Press, 1988.

[92] BC Goodwin. An entrainment model for timed enzyme syntheses in bacteria. *Nature*, 209(5022):479–481, 1966.

[93] Stephen Jay Gould. *The structure of evolutionary theory*. Harvard University Press, 2002.

[94] Stephen Jay Gould and Niles Eldredge. Punctuated equilibria: an alternative to phyletic gradualism. *Models in Paleobiology*, pages 82–115, 1972.

[95] Robert M Gray. Toeplitz and circulant matrices: A review. *Foundations and Trends in Communications and Information Theory*, 2(3):155–239, 2006.

[96] Sam F Greenbury and Sebastian E Ahnert. The organization of biological sequences into constrained and unconstrained parts determines fundamental properties of genotype–phenotype maps. *Journal of The Royal Society Interface*, 12(113):20150724, 2015.

[97] Sam F Greenbury, Iain G Johnston, Ard A Louis, and Sebastian E Ahnert. A tractable genotype–phenotype map modelling the self-assembly of protein quaternary structure. *Journal of The Royal Society Interface*, 11(95):20140249, 2014.

[98] Sam F Greenbury, Iain G Johnston, Matthew A Smith, Jonathan PK Doye, and Ard A Louis. The effect of scale-free topology on the robustness and evolvability of genetic regulatory networks. *Journal of theoretical biology*, 267(1):48–61, 2010.

[99] Sam F Greenbury, Steffen Schaper, Sebastian E Ahnert, and Ard A Louis. Genetic correlations greatly increase mutational robustness and can both reduce and enhance evolvability. *PLoS computational biology*, 12(3):e1004773, 2016.

[100] Liza Gross. Are "ultraconserved" genetic elements really indispensable? *PLoS biology*, 5(9):e253, 2007.

[101] Ryan N Gutenkunst, Jordan C Atlas, Fergal P Casey, Brian C Daniels, Robert S Kuczenski, Joshua J Waterfall, Chris R Myers, and James P Sethna. Sloppycell. *See http://sloppycell.sourceforge.net*, 2007.

[102] Ryan N Gutenkunst, Joshua J Waterfall, Fergal P Casey, Kevin S Brown, Christopher R Myers, and James P Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS computational biology*, 3(10):e189, 2007.

[103] John BS Haldane. A mathematical theory of natural and artificial selection. *Mathematical Proceedings of the Cambridge Philosophical Society*, 23(04):363, oct 1926.

[104] Baris Hancioglu and John J Tyson. A mathematical model of mitotic exit in budding yeast: the role of polo kinase. *PLOS ONE*, 7(2):e30810, 2012.

[105] Tomáš Helikar, John Konvalina, Jack Heidel, and Jim A Rogers. Emergent decision-making in biological signal transduction networks. *Proceedings of the National Academy of Sciences*, 105(6):1913–1918, 2008.

[106] I.L. Hofacker, W. Fontana, P.F. Stadler, L.S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125(2):167–188, 1994.

[107] John J Hopfield and David W Tank. Collective computation with continuous variables. In *Disordered Systems and Biological Organization*, pages 155–170. Springer, 1986.

[108] John J Hopfield and David W Tank. Computing with neural circuits - a model. *Science*, 233(4764):625–633, 1986.

[109] Cristián Huepe and Maximino Aldana-González. Dynamical phase transition in a neural network model with noise: an exact solution. *Journal of Statistical Physics*, 108(3):527–540, 2002.

[110] Julian Huxley. *Evolution the modern synthesis*. George Allen and Unwin, 1942.

[111] Esther Ibáñez-Marcelo and Tomás Alarcón. The topology of robustness and evolvability in evolutionary systems with genotype–phenotype map. *Journal of theoretical biology*, 356:144–162, 2014.

[112] Thomas Jörg, Olivier C Martin, and Andreas Wagner. Neutral network sizes of biological RNA molecules can be computed and are not atypically small. *BMC bioinformatics*, 9(1):464, 2008.

[113] Leo Kadanoff, Susan Coppersmith, and Maximino Aldana. Boolean dynamics with random couplings. *arXiv preprint nlin/0204062*, 2002.

[114] Stuart A Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224(5215):177–178, 1969.

[115] Stuart A Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467, 1969.

[116] Stuart A Kauffman. The large scale structure and dynamics of gene control circuits: an ensemble approach. *Journal of Theoretical Biology*, 44(1):167–190, 1974.

[117] Stuart A Kauffman. Boolean systems, adaptive automata, evolution. *Disordered systems and biological organization*, pages 339–360, 1986.

[118] Stuart A Kauffman. A framework to think about evolving genetic regulatory systems. *Integrating Scientific Disciplines: Case Studies from the Life Sciences*, 2:165, 1986.

[119] Stuart A Kauffman. *The Origins of Order: Self-organization and selection in evolution*. Oxford University Press, USA, 1993.

[120] Stuart A Kauffman. *At Home in the Universe: The search for the laws of self-organization and complexity*. Oxford University Press, 1996.

[121] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[122] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[123] Boris N Kholodenko. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *The FEBS Journal*, 267(6):1583–1588, 2000.

[124] Motoo Kimura. Evolutionary rate at the molecular level. *Nature*, 217(5129):624–626, 1968.

[125] Motoo Kimura. *The neutral theory of molecular evolution*. Cambridge University Press, 1983.

[126] Andrei Nikolaevich Kolmogorov. Three approaches to the definition of the concept "quantity of information". *Problemy peredachi informatsii*, 1(1):3–11, 1965.

[127] Leon Gordon Kraft. *A device for quantizing, grouping, and coding amplitude-modulated pulses*. PhD thesis, Massachusetts Institute of Technology, 1949.

[128] Pavel Kraikivski, Katherine C Chen, Teeraphan Laomettachit, TM Murali, and John J Tyson. From start to finish: computational analysis of cell cycle control in budding yeast. *Npj Systems Biology And Applications*, 1:15016, 2015.

[129] Karl E Kürten. Critical phenomena in model neural networks. *Physics Letters A*, 129(3):157–160, 1988.

[130] Kevin Laland, Gregory A Wray, and Hopi E Hoekstra. Does evolutionary theory need a rethink? *Nature*, 514(7521):161, 2014.

[131] 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. In *Proc. R. Soc. B*, volume 282, page 20151019. The Royal Society, 2015.

[132] David S Latchman. Transcription factors: an overview. *The international journal of biochemistry & cell biology*, 29(12):1305–1312, 1997.

[133] Kai-Yeung Lau, Surya Ganguli, and Chao Tang. Function constrains network architecture and dynamics: a case study on the yeast cell cycle Boolean network. *Physical Review E*, 75(5):051907, 2007.

[134] Kit Fun Lau and Ken A Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10):3986–3997, 1989.

[135] Robert D Leclerc. Survival of the sparsest: robust gene networks are parsimonious. *Molecular systems biology*, 4(1):213, 2008.

[136] Ethan Lee, Adrian Salic, Roland Krüger, Reinhart Heinrich, and Marc W Kirschner. The roles of APC and Axin derived from experimental and theoretical analysis of the Wnt pathway. *PLoS biology*, 1(1):e10, 2003.

[137] Jean-Christophe Leloup and Albert Goldbeter. Chaos and birhythmicity in a model for circadian oscillations of the PER and TIM proteins in Drosophila. *Journal of theoretical biology*, 198(3):445–459, 1999.

[138] Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22(1):75–81, 1976.

[139] Annick Lesne. Shannon entropy: a rigorous notion at the crossroads between probability, information theory, dynamical systems and statistical physics. *Mathematical Structures in Computer Science*, 24(03):e240311, 2014.

[140] Annick Lesne, Jean-Luc Blanc, and Laurent Pezard. Entropy estimation of very short symbolic sequences. *Physical Review E*, 79(4):046208, 2009.

[141] Tim Leung and Xin Li. *Optimal Mean Reversion Trading: Mathematical Analysis and Practical Applications*, volume 1. World Scientific, 2015.

[142] Leonid Anatolevich Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problemy Peredachi Informatsii*, 10(3):30–35, 1974.

[143] Leonid Anatolevich Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

[144] Richard C Lewontin. *The genetic basis of evolutionary change*, volume 560. Columbia University Press New York, 1974.

[145] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences of the United States of America*, 101(14):4781–4786, 2004.

[146] Hao Li, Robert Helling, Chao Tang, and Ned Wingreen. Emergence of preferred structures in a simple model of protein folding. *Science*, 273(5275):666, 1996.

[147] Ming Li, Jonathan H Badger, Xin Chen, Sam Kwong, Paul Kearney, and Haoyong Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154, 2001.

[148] Ming Li and Paul MB Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer Publishing Company, Incorporated, 2008.

[149] Song Li, Sarah M Assmann, and Réka Albert. Predicting essential components of signal transduction networks: a dynamic model of guard cell abscisic acid signaling. *PLoS biology*, 4(10):e312, 2006.

[150] Jefrey Lijffijt, Panagiotis Papapetrou, Jaakko Hollmén, and Vassilis Athitsos. Benchmarking dynamic time warping for music retrieval. In *Proceedings of the 3rd international conference on pervasive technologies related to assistive environments*, page 59. ACM, 2010.

[151] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.

[152] Aristid Lindenmayer. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3):280–299, 1968.

[153] Seth Lloyd. Quantum-mechanical computers and uncomputability. *Physical review letters*, 71(6):943, 1993.

[154] James CW Locke, Megan M Southern, László Kozma-Bognár, Victoria Hibberd, Paul E Brown, Matthew S Turner, and Andrew J Millar. Extension of a genetic

network model by iterative experimentation and mathematical analysis. *Molecular systems biology*, 1(1), 2005.

[155] Jason G Lomnitz and Michael A Savageau. Elucidating the genotype–phenotype map by automatic enumeration and analysis of the phenotypic repertoire. *NPJ systems biology and applications*, 1, 2015.

[156] Anna Lovrics, Attila Csikász-Nagy, István Gy Zsély, Judit Zádor, Tamás Turányi, and Béla Novák. Time scale and dimension analysis of a budding yeast cell cycle model. *BMC bioinformatics*, 7(1):494, 2006.

[157] Michael Lynch. The evolution of genetic networks by non-adaptive processes. *Nature reviews. Genetics*, 8(10):803, 2007.

[158] Siddharth Madan and Kristin J Dana. Modified balanced iterative reducing and clustering using hierarchies (m-BIRCH) for visual clustering. *Pattern Analysis and Applications*, 19(4):1023–1040, 2016.

[159] S Mandal, G Saha, and RK Pal. A survey on recurrent neural network based modelling of gene regulatory network. *MOJ Proteomics Bioinformatics*, 4(3):00125, 2016.

[160] Susanna Manrubia and José A Cuesta. Evolution on neutral networks accelerates the ticking rate of the molecular clock. *Journal of The Royal Society Interface*, 12(102):20141010, 2015.

[161] Susanna Manrubia and José A Cuesta. Distribution of genotype network sizes in sequence-to-structure genotype–phenotype maps. *Journal of The Royal Society Interface*, 14(129):20160976, 2017.

[162] Susanna C Manrubia and Jose A Cuesta. Neutral networks of genotypes: Evolution behind the curtain. *arXiv preprint arXiv:1002.2745*, 2010.

[163] LC Martin, Gregory B Gloor, SD Dunn, and Lindi M Wahl. Using information theory to search for co-evolving residues in proteins. *Bioinformatics*, 21(22):4116–4124, 2005.

[164] Per Martin-Löf. The definition of random sequences. *Information and control*, 9(6):602–619, 1966.

[165] Stefano Martiniani, K Julian Schrenk, Jacob D Stevenson, David J Wales, and Daan Frenkel. Turning intractable counting into sampling: Computing the configurational entropy of three-dimensional jammed packings. *Physical Review E*, 93(1):012906, 2016.

[166] Stefano Martiniani, Jacob D Stevenson, David J Wales, and Daan Frenkel. Superposition enhanced nested sampling. *Physical Review X*, 4(3):031034, 2014.

[167] J Matoušek and J Nešetřil. *Invitation to Discrete Mathematics*. Oxford University Press, 2009.

[168] Robert M May. Models for two interacting populations. *Theoretical ecology: principles and applications*, pages 49–70, 1976.

[169] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[170] Cory Y McLean, Philip L Reno, Alex A Pollen, Abraham I Bassan, Terence D Capellini, Catherine Guenther, Vahan B Indjeian, Xinhong Lim, Douglas B Menke, and Bruce T Schaar. Human-specific loss of regulatory DNA and the evolution of human-specific traits. *Nature*, 471(7337):216, 2011.

[171] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, USA, 2009.

[172] Tamara Mihaljev and Barbara Drossel. Scaling in a general class of critical random Boolean networks. *Physical Review E*, 74(4):046101, 2006.

[173] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[174] Gerd B Müller. Evo-devo: extending the evolutionary synthesis. *Nature reviews. Genetics*, 8(12):943, 2007.

[175] Christopher R Myers, Ryan N Gutenkunst, and James P Sethna. Python unleashed on systems biology. *Computing in Science & Engineering*, 9(3):34–37, 2007.

[176] Geeta J Narlikar, Hua-Ying Fan, and Robert E Kingston. Cooperation between complexes that regulate chromatin structure and transcription. *Cell*, 108(4):475–487, 2002.

[177] Denis Noble. Evolution beyond neo-Darwinism: a new conceptual framework. *Journal of Experimental Biology*, 218(1):7–13, 2015.

[178] Yigal D Nochomovitz and Hao Li. Highly designable phenotypes and mutational buffers emerge from a systematic mapping between network topology and dynamic output. *Proceedings of the National Academy of Sciences of the United States of America*, 103(11):4180–4185, 2006.

[179] Béla Novák and John J Tyson. Design principles of biochemical oscillators. *Nature reviews. Molecular cell biology*, 9(12):981, 2008.

[180] K Ochiai, T Yamanaka, K Kimura, and O Sawada. Inheritance of drug resistance (and its transfer) between shigella strains and between Shigella and E. coli strains. *Hihon Iji Shimpor,(in Japanese)*, 34:1861, 1959.

[181] Thomas E Ouldridge. The importance of thermodynamics for molecular systems, and the importance of molecular systems for thermodynamics. *arXiv preprint arXiv:1702.00360*, 2017.

[182] Joshua L Payne, Jason H Moore, and Andreas Wagner. Robustness, evolvability, and the logic of genetic regulation. *Artificial life*, 20(1):111–126, 2014.

[183] Joshua L Payne and Andreas Wagner. Constraint and contingency in multifunctional gene regulatory circuits. *PLoS computational biology*, 9(6):e1003071, 2013.

[184] Joshua L Payne and Andreas Wagner. Latent phenotypes pervade gene regulatory circuits. *BMC systems biology*, 8(1):64, 2014.

[185] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[186] Tiago P Peixoto. The graph-tool python library. *figshare*, 2014.

[187] Sriram Pemmaraju and Steven Skiena. Implementing discrete mathematics: Combinatorics and graph theory with Mathematica, 2003.

[188] Charles S Peskin, Garrett M Odell, and George F Oster. Cellular motions and thermal fluctuations: the brownian ratchet. *Biophysical journal*, 65(1):316–324, 1993.

[189] Massimo Pigliucci and Leonard Finkelman. The extended (evolutionary) synthesis debate: where science meets philosophy. *BioScience*, 64(6):511–516, 2014.

[190] Massimo Pigliucci and Gerd B. Müller, editors. *Evolution—the Extended Synthesis*. The MIT Press, mar 2010.

[191] Marjan Kuchaki Rafsanjani, Zahra Asghari Varzaneh, and Nasibeh Emami Chukanlo. A survey of hierarchical clustering algorithms. *The Journal of Mathematics and Computer Science*, 5(3):229–240, 2012.

[192] Karthik Raman and Andreas Wagner. The evolvability of programmable hardware. *Journal of The Royal Society Interface*, page rsif20100212, 2010.

[193] Karthik Raman and Andreas Wagner. Evolvability and robustness in a complex signalling circuit. *Molecular BioSystems*, 7(4):1081–1092, 2011.

[194] Elisabeth Remy, Paul Ruet, Luis Mendoza, Denis Thieffry, and Claudine Chaouiya. From logical regulatory graphs to standard petri nets: dynamical roles and functionality of feedback circuits. In *Transactions on Computational Systems Biology VII*, pages 56–72. Springer, 2006.

[195] Eric Rivals, Olivier Delgrange, J-P Delahaye, Max Dauchet, M-O Delorme, Alain Hénaut, and Emmanuelle Ollivier. Detection of significant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences. *Bioinformatics*, 13(2):131–136, 1997.

[196] Thimo Rohlf. Critical line in random-threshold networks with inhomogeneous thresholds. *Physical Review E*, 78(6):066118, 2008.

[197] Thimo Rohlf and Stefan Bornholdt. Criticality in random threshold networks: annealed approximation and beyond. *Physica A: Statistical Mechanics and its Applications*, 310(1):245–259, 2002.

[198] Marc R Roussel and Rui Zhu. Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. *Physical biology*, 3(4):274, 2006.

[199] Lynn Sagan. On the origin of mitosing cells. *Journal of theoretical biology*, 14(3):225IN1–274IN6, 1967.

[200] Yasushi Sakurai, Masatoshi Yoshikawa, and Christos Faloutsos. Ftw: fast similarity search under the time warping distance. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 326–337. ACM, 2005.

[201] Satoru Sasagawa, Yu-ichi Ozaki, Kazuhiro Fujita, and Shinya Kuroda. Prediction and validation of the distinct dynamics of transient and sustained ERK activation. *Nature cell biology*, 7(4):365–373, 2005.

[202] Steffen Schaper and Ard A Louis. The arrival of the frequent: how bias in genotype-phenotype maps can steer populations to local optima. *PLOS ONE*, 9(2):e86635, 2014.

[203] Thomas D Schneider. A brief review of molecular information theory. *Nano communication networks*, 1(3):173–180, 2010.

[204] Aaron W Schrey, Christina L Richards, Victoria Meller, Vincent Sollars, and Douglas M Ruden. The role of epigenetics in evolution: the extended synthesis. *Genetics research international*, 2012, 2012.

[205] Marcel P Schützenberger. Algorithms and the neo-Darwinian theory of evolution. In *The Wistar Institute symposium monograph*, volume 5, page 73, 1967.

[206] Wolfgang Schumann. *Dynamics of the Bacterial Chromosome: Structure and Function*. John Wiley & Sons, 2006.

[207] Peter Schuster. Prediction of RNA secondary structures: from theory to models and real molecules. *Reports on Progress in Physics*, 69(5):1419, 2006.

[208] Peter Schuster, Walter Fontana, Peter F Stadler, and Ivo L Hofacker. From sequences to shapes and back: a case study in RNA secondary structures. *Proceedings of the Royal Society of London B: Biological Sciences*, 255(1344):279–284, 1994.

[209] Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64, 2002.

[210] Adam A Smith and Mark Craven. Fast multisegment alignments for temporal expression profiles. In *Computational Systems Bioinformatics/Life Sciences Society. Computational Systems Bioinformatics Conference*, volume 7, page 315. NIH Public Access, 2008.

[211] John Maynard Smith. Natural selection and the concept of a protein space. *Nature*, 225(5232):563–564, 1970.

[212] Fernando Soler-Toscano, Hector Zenil, Jean-Paul Delahaye, and Nicolas Gauvrit. Calculating Kolmogorov complexity from the output frequency distributions of small Turing machines. *PLOS ONE*, 9(5):e96223, 2014.

[213] Ray J Solomonoff. A preliminary report on a general theory of inductive inference. *Cambridge, MA: Zator Co*, 1960.

[214] Christopher F Steiner. Environmental noise, genetic diversity and the evolution of evolvability and robustness in model gene networks. *PLOS ONE*, 7(12):e52204, 2012.

[215] Kevin Struhl. Fundamentally different logic of gene regulation in eukaryotes and prokaryotes. *Cell*, 98(1):1–4, 1999.

[216] Robert H Swendsen and Jian-Sheng Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57(21):2607, 1986.

[217] Agnes Szejka, Tamara Mihaljev, and Barbara Drossel. The phase diagram of random threshold networks. *New Journal of Physics*, 10(6):063009, 2008.

[218] Tianhai Tian and Kevin Burrage. Stochastic neural network models for gene regulatory networks. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, pages 162–169. IEEE, 2003.

[219] Gašper Tkačik and William Bialek. Information processing in living systems. *Annual Review of Condensed Matter Physics*, 7:89–117, 2016.

[220] Christian Tönsing, Jens Timmer, and Clemens Kreutz. Cause and cure of sloppiness in ordinary differential equation models. *Physical Review E*, 90(2):023303, 2014.

[221] Christian Torres-Sosa, Sui Huang, and Maximino Aldana. Criticality is an emergent property of genetic networks that exhibit evolvability. *PLoS computational biology*, 8(9):e1002669, 2012.

[222] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58(345-363):5, 1936.

[223] John J Tyson. Modeling the cell division cycle: cdc2 and cyclin interactions. *Proceedings of the National Academy of Sciences*, 88(16):7328–7332, 1991.

[224] John J Tyson, Reka Albert, Albert Goldbeter, Peter Ruoff, and Jill Sible. Biological switches and clocks. *Journal of The Royal Society Interface*, 5(Suppl_1):S1–S8, aug 2008.

[225] John J Tyson, Kathy Chen, and Bela Novak. Network dynamics and cell physiology. *Nature Reviews Molecular Cell Biology*, 2(12):908–916, 2001.

[226] Hiroki R Ueda, Masatoshi Hagiwara, and Hiroaki Kitano. Robust oscillations within the interlocked feedback model of Drosophila circadian rhythm. *Journal of theoretical biology*, 210(4):401–406, 2001.

[227] Leslie Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books (AZ), 2013.

[228] Vera Van Noort, Berend Snel, and Martijn A Huynen. The yeast coexpression network has a small-world, scale-free architecture and can be explained by a simple model. *EMBO reports*, 5(3):280–284, 2004.

[229] J.M.G. Vilar, H.Y. Kueh, N. Barkai, and S. Leibler. Mechanisms of noise-resistance in genetic oscillators. *Proceedings of the National Academy of Sciences of the United States of America*, 99(9):5988, 2002.

[230] Paul MB Vitányi. Similarity and denoising. *Phil. Trans. R. Soc. A*, 371(1984), 2013.

[231] Jiří Vohradský. Neural network model of gene expression. *The FASEB journal*, 15(3):846–854, 2001.

[232] Andreas Wagner. Robustness and evolvability: a paradox resolved. *Proceedings of the Royal Society of London B: Biological Sciences*, 275(1630):91–100, 2008.

[233] Andreas Wagner. *Robustness and evolvability in living systems*. Princeton University Press, 2013.

[234] Andreas Wagner. *Arrival of the Fittest: Solving Evolution's Greatest Puzzle*. Penguin, 2014.

[235] Günter P Wagner and Manfred D Laubichler. Rupert Riedl and the re-synthesis of evolutionary and developmental biology: body plans and evolvability. *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, 302(1):92–102, 2004.

[236] Mitchell M Waldrop. *Complexity: The emerging science at the edge of order and chaos*. Simon and Schuster, 1993.

[237] Jean-Charles Walter and Gerard T Barkema. An introduction to Monte Carlo methods. *Physica A: Statistical Mechanics and its Applications*, 418:78–87, 2015.

[238] Guanyu Wang, Chenghang Du, Hao Chen, Rahul Simha, Yongwu Rong, Yi Xiao, and Chen Zeng. Process-based network decomposition reveals backbone motif structure. *Proceedings of the National Academy of Sciences*, 107(23):10478–10483, 2010.

[239] Joshua J Waterfall, Fergal P Casey, Ryan N Gutenkunst, Kevin S Brown, Christopher R Myers, Piet W Brouwer, Veit Elser, and James P Sethna. Sloppy-model universality class and the Vandermonde matrix. *Physical review letters*, 97(15):150601, 2006.

[240] Richard A Watson and Eörs Szathmáry. How can evolution learn? *Trends in ecology & evolution*, 31(2):147–157, 2016.

[241] Marcel Weiß and Sebastian E Ahnert. Phenotypes can be robust and evolvable if mutations have non-local effects on sequence constraints. *Journal of The Royal Society Interface*, 15(138):20170618, 2018.

[242] Karen Willbrand, Francois Radvanyi, Jean-Pierre Nadal, Jean-Paul Thiery, and Thomas MA Fink. Identifying genes from up–down properties of microarray expression series. *Bioinformatics*, 21(20):3859–3864, 2005.

[243] Sewall Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. *Proceedings of the Sixth International Congress of Genetics, Ithaca, New York, 1932*, pages 356–366, 1932.

[244] Andrew Wuensche. Attractor basins of discrete networks. *Cognitive Science Research Paper*, 461, 1997.

[245] Daniel E Zak, Gregory E Gonye, James S Schwaber, and Francis J Doyle. Importance of input perturbations and stochastic gene expression in the reverse engineering of genetic regulatory networks: insights from an identifiability analysis of an *in silico* network. *Genome research*, 13(11):2396–2405, 2003.

[246] Jorge GT Zañudo, Maximino Aldana, and Gustavo Martínez-Mekler. Boolean threshold networks: Virtues and limitations for biological modeling. *Information Processing and Biological Systems*, pages 113–151, 2011.

[247] Hector Zenil and Jean-Paul Delahaye. An algorithmic information theoretic approach to the behaviour of financial markets. *Journal of Economic Surveys*, 25(3):431–463, 2011.

[248] Hector Zenil, Fernando Soler-Toscano, Jean-Paul Delahaye, and Nicolas Gauvrit. Two-dimensional Kolmogorov complexity and an empirical validation of the coding theorem method by compressibility. *PeerJ Computer Science*, 1:e23, 2015.

[249] Hector Zenil, Fernando Soler-Toscano, Kamaludin Dingle, and Ard A Louis. Correlation of automorphism group size and topological properties with program-size complexity evaluations of graphs and complex networks. *Physica A: Statistical Mechanics and its Applications*, 404:341–358, 2014.

[250] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, number 2, pages 103–114. ACM, 1996.

[251] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[252] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

[253] Jason W Zwolak, John J Tyson, and Layne T Watson. Globally optimised parameters for a model of mitotic control in frog egg extracts. *IEE Proceedings-Systems Biology*, 152(2):81–92, 2005.