Imperial College London

BEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

Evolving Promoter Architectures for Encoding Extracellular Information within Budding Yeast

Author:
James Matthew Uygongco Young

Supervisor:
Dr Vahid Shahrezaei

June 21, 2023

Submitted in partial fulfilment of the requirements
for the BEng in Mathematics and Computer Science of Imperial College London

Abstract

When subjected to stress, yeast cells can adapt through some intracellular decision-making involving the activation and repression of its genes. We model the activity in the promoter region responsible for toggling gene activity as time-inhomogeneous continuous-time Markov chains and aim to search for architectures whose trajectories efficiently encode the information about a cell's environment. To do this, we build a pipeline for evaluating models. We propose a matrix exponential approximation to simulate trajectories, which are then estimated by a machine-learning decoder of their mutual information with the environment. We employ a genetic algorithm framework and show it can navigate through the combinatorically large space of models to search for fit solutions, most of which exhibit a chain-like architecture with edges linearly dependent on pairs of transcription factors (TFs) with complementary nuclear traces. This structure acts as a noise-suppression mechanism, with the chain's length controlling the lag before switching active. Furthermore, evolution under discrete and continuous paradigms of gene activity found the two to have comparable fitnesses with simpler structures for the continuous case. As populations converge to similar architectures under high selective pressure, distance metrics are formulated between promoter models so a quality-diversity algorithm can produce a collection of fit and diverse solutions. These are analysed of their structure, and indeed we find the evolutionary search's affinity towards pairing complementary TFs and having models' average trajectories resemble a TF's nuclear traces.

Acknowledgements

Behind this paper is an outpouring of support I would like to recognize.

To Dr Vahid Shahrezaei, my supervisor, for guiding me whilst remaining optimistic in my every progress. To Dr Peter Swain from the University of Edinburgh for his valuable technical input and expertise. To Kamil for his previous work on the topic.

This one's for my mom and her strength in raising three boys. This one's for my late dad who is surely excited to see me graduate. This one's for those who keep going when all else fails. To friends who listened. To the mochas that kept me going. To the first of many more.

Contents

Chapter 1

Introduction

Genes, at a fundamental level, act as a compilation of nature's own assembly language—DNA. Stored within them are instructions for synthesising proteins involved in cellular function. To initiate these instructions and thereby express the gene, proteins called *transcription factors* (TFs) must bind to specific promoter regions in DNA [1]. This course of action is stochastic [2] and sets the central dogma in motion. By expressing or repressing genes, cells can respond and adapt to stimuli, including environmentally-induced stress [3], paving the way for real-time cell decision-making. However, the exact ways in which a cell encodes its environment through TF signals remain nontrivial due to its spatial and temporal dynamics [4] and noise from molecular interactions [2, 5].

Eukaryotic cells are distinct for housing their genes within a membrane called the nucleus [1]. Inside this membrane, a measurable shift in the concentration of TFs can occur in the event of stress, making eukaryotes preferable for analysis. While many studies have investigated the dynamics of nuclear translocation [6, 7] and signalling pathways [8], the inference of combinatoric TF encodings [9] and development of transcription bursting models [10, 11] persist to be of interest. As a result, we turn to recent practical measurements to support our judgement in developing models.

Saccharomyces cerevisiae, the budding yeast, is an extensively studied archetype of the eukaryotic cell. In a seminal paper, Granados et al. [12] performed single-cell microscopy experiments to quantify how extracellular information, namely stress, could be organised intracellularly within *S. cerevisiae*. By treating their measurements as an exogenous input, models that capture the stochasticity of gene expression can be examined and optimised for high mutual information between the environment and the expressed gene. In doing so, an inherent TF-signalling structure can be suggested that best reflects existing data. But, more importantly, relationships between the dynamics and noise of TFs can be identified.

A previous master's thesis [13] set the groundwork with promoter models, establishing hypotheses on potential cooperative-like scenarios. However, simulation bottlenecks limited the scope of analyses, and generalisability remains challenging due to the combinatorial explosion of models.

1.1 Objectives

This study aims twofold: exercise a methodology that resolves known bottlenecks and allows further analysis; consider more prominent but previously intractable extensions. Specifically, we aim to hasten the evaluation of promoter models and provide meaningful insights into the role of TFs in transcription regulation. To further the latter, we aim to build a diverse collection of effective architectures from the combinatorically large search space.

The crux of our method lies within the model evaluation pipeline: a stochastic simulation to generate sample trajectories for promoter activity and an estimation of the mutual information it produces. We remark our focus on the gene's activity rather than its mRNA production.

1.2 Challenges

Due to the multifaceted nature of the project, there were several moving parts with many degrees of freedom. As a result, most of the work was spent on validating results and benchmarking design choices. Below, we list some challenges faced in this light.

- **Memory and time trade-offs**: Often, calculations were batched in tensors for efficiency reasons. However, as with matrix exponentials, a vectorised approach produced quicker execution times at the cost of a much larger memory footprint. While this was not an issue on a reasonable number of models, it became problematic with larger runs on an HPC cluster where jobs have a fixed time and memory allocation. In addition, searching for memory leaks and optimisations on multi-processing code was challenging as jobs were terminated without a meaningful trace when such constraints were violated. Through memory profiling tests, certain thresholds were put in place for which an algorithm would default to a slower but less memory-intensive approach.

- **Parameters for the evolutionary search**: Orchestrating a successful evolutionary search involves rigorous testing of many hyperparameters. For instance, the genetic algorithm alone required the tweaking of frequencies of up to eight mutation operators to avoid premature convergence and the choice of schemes that provide appropriate selective pressure. From the novelty search aspect, various archival strategies were experimented with. Different novelty metrics define spaces with possibly different dimensionalities, requiring careful consideration of archival thresholds or densities. We acknowledge the amount of parameter tuning necessary as a weakness of our method.

- **Efficient nearest neighbour search for arbitrary metric spaces**: Space-partitioning data structures for metric spaces mostly rely on the triangle inequality to prune the search queries. As such, diminishing returns are commonplace for high-dimensional metric spaces. Vantage point trees, M-trees, and cover trees were implemented to aid in nearest neighbour searches, but the number of calls to the distance metric often resulted in $O(n^2)$ complexity. As such, a brute-force approach taking symmetry into account was ultimately superior. Approximate nearest neighbours were also briefly explored. However, with our unconventional metric space, no assumptions about the fitness around a model's locality could be made to guarantee that local competition scores were accurate.

1.3 Contributions

While the pipeline was originally the project's focal point, we also proposed new ideas to aid in the analysis of models which can be reused in other contexts. Below, we summarise what we believe to be notable takeaways from the study in chronological order.

- We frame promoter dynamics as time-inhomogeneous continuous-time Markov chains and demonstrate their capability of producing trajectories that encode different forms of environmental stress. We further consider three model paradigms of increasing generality and show how they differ when optimised.
- We find a good incentive to decouple mRNA from the system and propose an approximation by iteratively multiplying matrix exponentials, yielding similar results to exact SSA methods. We show it is highly vectorisable in computation, has deterministic simulation times, and is adaptable to compute the average trend of the entire population efficiently. We improve exact SSA efforts on the order of 1×10^3 times for four-state models and several higher orders of magnitude for larger models.
- We find a trade-off in speed and accuracy to keep simulation and decoding times within the same order of magnitude to design a high throughput evaluation pipeline.
- We introduce a genetic algorithm framework for evolving promoter models with constraint-preserving genetic operators, e.g. our subgraph-swapping crossover. We show it can find models with exceptionally high MI and tackles the combinatorically large space well, optimising a model's architecture and weights at the same time.
- We present a formulation of models as points in a metric space under two proposed distance metrics: a topology measure based on the Wasserstein Weisfeiler-Lehman scheme and a trajectory measure that differentiates between average activity trends, computable from our approximation method. We also show which aspects of a model each metric discriminates against to drive the search.
- Coupled with the above metrics, we perform a novelty search with local competition under an unstructured quality-diversity archival strategy and elitism as in NSGA-II to draw similarities between good models through the archive. This can be generalised for evolving diverse graphs evaluated under black-box fitness functions.
- We publish the code on a public repository[1] with documented examples for benchmarks, analysis of models and evolutionary search trends, and various visualisations.

1.4 Ethical Considerations

We use Imperial College's High-Performance Computing cluster to run large memory and processor-heavy jobs. We recognize its large carbon footprint and only run jobs when necessary for this study.

S. cerevisiae is widely used in the brewing industry. As such, any information on the budding yeast can, in some form, lead to malpractice. The extent to which this study contributes, however, is limited.

Single-cell microscopy data comes from a previous study following appropriate guidelines

[1]The repository is accessible at github.com/matsagad/beng-project.

and laboratory etiquette. As such, no living organisms are directly involved in this study. Furthermore, the data is under a Creative Commons Attribution 4.0 International license. We have permission from the authors and provide proper attribution.

Chapter 2

Preliminaries

In this chapter, we introduce material fundamental to the methods undertaken in the study. These concepts are presented together with their biological counterparts for the reader to develop an appreciation for the subject matter.

We first establish the setting on the genome level with the role of transcription factors. We then introduce a class of stochastic processes and assess a simple model for transcription. Lastly, we present tools from classical information theory and show how environmental stress can act as quantifiable signals for cell decision-making.

2.1 Biological Underpinnings

A somewhat counterintuitive fact is that virtually all cells in the human body contain the same piece of genetic information. One might then ask: how can cells differentiate their function? The answer is **gene regulation**. This act of regulating gene expression has been extensively studied ever since the widely adopted central dogma of biology was proposed by Crick [14]. In it, the flow of genetic information is described to move from DNA to RNA, the transcription stage, and from RNA to protein, the translation stage. Gene regulation is mainly a byproduct of complex networks and feedback loops induced by these two stages. We focus on transcription, the first of the two, and briefly discuss its course of action.

2.1.1 Transcription Factor Dynamics

The initiation of the transcription process involves proteins called **transcription factors** (**TFs**) to bind to specific promoter regions in DNA [15]. Generally, TFs can be classified as activators or repressors depending on whether they initiate or inhibit the expression of certain genes. For example, when yeast cells undergo oxidative stress, a signalling pathway is triggered, leading to activator TFs **Msn2** and **Msn4** to accumulate in the nucleus and admit an adaptive response [16].

Combinations of TFs allow for a coordinated expression of various genes. Another source of complex behaviour is attributed to feedback loops wherein a protein synthesised by the expression of one gene acts as a TF for another [17]. These interconnected relationships form what is known as a gene regulatory network [18]. In the case of budding yeast, the number of known TFs, at least over a hundred [19], is viable enough for individual analysis.

However, with at least 2^{100} subsets, their combinatorial pairings are not. Inference for gene regulatory networks, therefore, persists to be a difficult task.

In eukaryotic cells, measurements for TF binding typically rely on their variable concentration or lack thereof within the nucleus. A natural inclination is to believe that genes are activated to varying degrees. Certainly, while population averaging allows for a continuous range of gene activity, on the scale of a single cell, the activity of genes and the biochemical reactions they depend on are discrete [20]. This intrinsic noise is why transcription, like most cellular activities, is deemed stochastic [20, 2].

Stochastic models have henceforth been considered to study this phenomenon.

2.2 Stochastic Processes

A **stochastic process** $(X_t)_{t\in\mathcal{T}}$ is a collection of random variables under a probability space for some time domain $\mathcal{T}$ and state space E common to all random variables. We consider a continuous time domain, $\mathcal{T} = \mathbb{R}\cup\{0\}$, and a discrete state space throughout the study.

2.2.1 Poisson Process

One of the most fundamental stochastic processes is the **Poisson counting process**, which takes values from $E = \mathbb{N}_0$ and is strictly non-decreasing. We say $(N_t)_{t\geq 0}$ is a Poisson process with rate $\lambda > 0$ if it satisfies [21]:

1. **Zero initial count**: $N_0 = 0$
2. **Independent increments**: Given any $n \in \mathbb{N}$ and $0 \leq t_0 < t_1 < \cdots < t_n$, the random variables $N_{t_0}, N_{t_1} - N_{t_0}, \ldots, N_{t_n} - N_{t_{n-1}}$ are independent.
3. **Stationary increments**: Given any two times $0 \leq s < t$ for any $k \in \mathbb{N}_0$,
$$\mathrm{P}(N_t - N_s = k) = \mathrm{P}(N_{t-s} = k)$$
4. **Single arrival times**: For any $t > 0$ and $0 < \delta \ll 1$,
$$\mathrm{P}(N_{t+\delta} - N_t = k) = \begin{cases} 1 - \lambda\delta + o(\delta) & k = 0 \\ \lambda\delta + o(\delta) & k = 1 \\ o(\delta) & k \geq 2 \end{cases}$$
where $o(\delta)$ is some function with $\lim_{\delta\to 0} o(\delta)/\delta = 0$.

Notably, the fourth definition above is in the form of its infinitesimal transition rates, which are similar in form to the Forward Kolmogorov equation governed by it. An equivalent statement is: for any $t \geq 0$, N_t follows a Poisson distribution $\mathrm{Poi}(\lambda t)$.

This is especially useful as the inter-arrival times H_i, i.e. the time between increments, follow an exponential distribution $\mathrm{Exp}(\lambda)$ and thus satisfy the lack of memory property. That is, given $x, y > 0$, H_i satisfies
$$\mathrm{P}(H_i > x + y \mid H_i > y) = \mathrm{P}(H_i > x)$$

A Poisson process is one of the simplest examples of a more general class of processes known as continuous-time Markov chains.

2.2.2 Continuous-Time Markov Chains

A **continuous-time Markov chain** (**CTMC**) is a stochastic process $(X_t)_{t\geq 0}$ with a continuous time domain taking values from some state space E which satisfies the **Markov property**. That is, given $n \in \mathbb{N}_0$, values $x_0, x_1, \ldots, x_n \in E$, and times $0 \leq t_0 < t_1 < \cdots < t_n$,

$$\mathrm{P}(X_{t_n} = x_n \mid X_{t_{n-1}} = x_{n-1}, \ldots, X_0 = x_0) = \mathrm{P}(X_{t_n} = x_n \mid X_{t_{n-1}} = x_{n-1})$$

Furthermore, a CTMC is said to be **time-homogenous** if it satisfies

$$\mathrm{P}(X_{t_{n+1}} = j \mid X_{t_n} = i) = \mathrm{P}(X_{t_{n+1}-t_n} = j \mid X_0 = i)$$

for $i, j \in E$ and $n \in \mathbb{N}_0$. In other words, the transition probabilities between states do not change with time.

2.2.2.1 Transitioning Between States

Markov chains with a discrete state space can be described as a set of states with some probability of moving between any two of them. CTMCs, like their discrete-time counterparts, can be represented by a **transition matrix** $\mathbf{P}_t = (p_{ij}(t))$ where $p_{ij}(t)$ is the probability of moving to state j at time t given the chain is in state i at time 0, i.e.

$$p_{ij}(t) = \mathrm{P}(X_t = j \mid X_0 = i)$$

Note that we assign $\mathbf{P}_0 = \mathbf{I}$. A CTMC thus begins in one state, and either remains or jumps to another state after time t has elapsed, according to its transition matrix $\mathbf{P}_t$.

Naturally, another way to frame CTMCs is by considering the holding times spent at each state. For $\delta > 0$, we define the generator matrix $\mathbf{Q} = (q_{ij})$ as

$$\mathbf{Q} = \lim_{\delta \downarrow 0} \frac{\mathbf{P}_\delta - \mathbf{P}_0}{\delta}$$

Denoting its non-diagonal entries as $q_{ij} = \lambda_{ij}$, the entries can be given by

$$q_{ij} = \begin{cases} \lambda_{ij}, & i \neq j \\ -\sum_{j \in E, j \neq i} \lambda_{ij}, & i = j \end{cases}$$

Suppose the chain is currently at state i and let $\mathcal{A}(i)$ be the set of states it can transition to. An analogous interpretation of simulating a CTMC is to set up exponential alarm clocks, one for each state in $\mathcal{A}(i)$. Here, an alarm clock corresponding to state j follows an exponential distribution $\mathrm{Exp}(\lambda_{ij})$. As soon as the first clock rings, the chain transitions to the state it belongs to, and the process is repeated.

Given $X_1, \ldots, X_n$ independent with $X_k \sim \mathrm{Exp}(\lambda_k)$, it can be shown that the minimum satisfies $H = \min\{X_k\} \sim \mathrm{Exp}(\sum_k \lambda_k)$ with

$$\mathrm{P}(H = X_j) = \frac{\lambda_j}{\sum_k \lambda_k}$$

When applied to our analogy, the above result can associate the holding times H_i with the probability of the chain moving to another state. The generator matrix is thus sufficient to capture the dynamics of the chain, but an expression for $\mathbf{P}_t$ is still often useful. It turns out that when $\{\mathbf{P}_t\}$ is a uniform semigroup, it can be expressed in terms of its generator.

2.2.2.2 Matrix Exponential Solution

Another property of Markov chains is that their transition matrix satisfies the Chapman-Kolmogorov equations, which in matrix form, is

$$\mathbf{P}_{t+s} = \mathbf{P}_t \mathbf{P}_s$$

An expression for the transition matrix can be derived from this. For $\delta > 0$,

$$\frac{\mathbf{P}_{t+\delta} - \mathbf{P}_t}{\delta} = \frac{\mathbf{P}_t(P_\delta - \mathbf{P}_0)}{\delta} = \mathbf{P}_t \frac{\mathbf{P}_\delta - \mathbf{P}_0}{\delta}$$

and taking the limit as $\delta \downarrow 0$,

$$\mathbf{P}'_t = \mathbf{P}_t \mathbf{Q}$$

The above differential equation is the **forward Kolmogorov equation** (**FKE**). In biological literature, it is also known as the **chemical master equation** with a similar but often transposed representation of matrices. Solving it admits the matrix exponential solution

$$\mathbf{P}_t = e^{t\mathbf{Q}}$$

2.2.2.3 Gene Activation Mechanism

Consider, as an example, a simple gene on-off switch determined by the rates equations

$$\mathrm{A} \xrightarrow{k_{off}} \mathrm{I}, \quad \mathrm{I} \xrightarrow{k_{on}} \mathrm{A}, \quad \mathrm{A} \xrightarrow{k_{syn}} \mathrm{A} + \mathrm{M}$$

where the gene switches between an active and inactive state given by rates k_{on} and k_{off}, and mRNA is produced with rate k_{syn} only when the gene is active. Denote n_X as the population of X. We then have the chemical master equations for the gene activity

$$\frac{dp_A}{dt} = -k_{off} p_A + k_{on} p_I \tag{2.1}$$

$$\frac{dp_I}{dt} = k_{off} p_A - k_{on} p_I \tag{2.2}$$

and the mRNA produced

$$\frac{dp_{M,0}}{dt} = -k_{syn} p_A p_{M,0}$$

$$\frac{dp_{M,i}}{dt} = k_{syn} p_A p_{M,i-1} - k_{syn} p_A p_{M,i}, \quad \text{for } i > 0$$

where $p_A = \mathrm{P}(n_A = 1, n_I = 0)$, $p_I = \mathrm{P}(n_A = 0, n_I = 1)$, and $p_{M,i} = \mathrm{P}(n_M = i)$.

Breaking down $p_A = p_{AA} + p_{AI}$ and $p_I = p_{IA} + p_{II}$ through Baye's rule and expressing 2.1 and 2.2 in matrix form, we find the FKE of an analogous two-state CTMC.

$$\frac{d}{dt}\begin{pmatrix} p_{AA} & p_{AI} \\ p_{IA} & p_{II} \end{pmatrix} = \begin{pmatrix} p_{AA} & p_{AI} \\ p_{IA} & p_{II} \end{pmatrix} \begin{pmatrix} -k_{off} & k_{off} \\ k_{on} & -k_{on} \end{pmatrix}$$

On the other hand, the second system is the FKE of a Poisson process with rate $\lambda = k_{syn} p_A$. We thus expect the amount of mRNA to follow such a process.

Simulating the first process yields a trajectory for the gene's activity over time. The next section discusses how information can be decoded from such time series.

2.3 Information Transmission

Due to the stochastic nature of molecular interactions and pathways for gene regulation, the quality of biochemical signals largely depends on their noise. Take environmental stress signals, for example. With increasing noise, the cell becomes unable to distinguish between its environmental identity and, as a result, can not adapt accordingly. We turn to classical information theory to formalise these concepts.

2.3.1 Classical Information Theory

In a 1984 landmark paper, Shannon [22] proposed a notion of entropy that eventually sprung forth the field of information theory and many of its other variations. **Shannon entropy**, or entropy for short, measures the uncertainty of a random variable X, taking values from a finite set $\mathcal{X}$. Denoting p_X as the probability density function of X, the entropy of X is defined as

$$\mathrm{H}(X) = \mathrm{E}\left[-\log_2(p_X(x))\right] = -\sum_{x\in\mathcal{X}} p_X(x)\log_2\left(p_X(x)\right)$$

An interpretation of this quantity is the unevenness of the probability distribution for X [23]. Particularly, it attains a minimum of zero when X has a deterministic value and a maximum of $\log_2\left(|\mathcal{X}|\right)$ when X has a uniform distribution across all values. Shannon [22] describes the logarithmic base as analogous to the unit for measuring information. Base-2, as with current computer hardware, lends the units of **bits**.

We note that the random variable is discrete and the sample space finite. A possible expression for the continuous case, known as differential entropy,

$$\mathrm{H}(X) = -\int_{\mathcal{X}} p_X(x)\log_2(p_X(x))dx$$

does not quite have as natural of an interpretation over the discrete case, notably differing as it permits negative values. Kolmogorov [24] proposed ϵ-entropy for arbitrary metric spaces to handle some of these concerns. However, this study only deals with the discrete case and directs the reader to some of his work.

From this formulation, a multivariate extension can be derived. Given random variables X and Y with finite sample spaces $\mathcal{X}$ and $\mathcal{Y}$ and p_{XY} as their joint probability density function, the **joint entropy** is given by

$$\mathrm{H}(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{XY}(x,y)\log_2\left(p_{XY}(x,y)\right)$$

and thus the **conditional entropy** can then be defined as

$$\mathrm{H}(X \mid Y) = \mathrm{H}(X,Y) - \mathrm{H}(Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{XY}(x,y)\log_2\left(\frac{p_{XY}(x,y)}{p_Y(y)}\right)$$

A related measure not to be confused with is **relative entropy**, which accounts for when X and Y share the same state space $\mathcal{X}$

$$\mathrm{H}_{rel}(X \mid Y) = -\sum_{x\in\mathcal{X}} p_X(x)\log_2\left(\frac{p_X(x)}{p_Y(x)}\right)$$

Taking its negation yields the Kullback-Liebler (KL) divergence, $d_{KL}(p_X \| p_Y)$, which has geometric interpretations for the two probability distributions.

Finally, we introduce **mutual information** (**MI**), given as [25]

$$\mathrm{I}(X;Y) = \mathrm{H}(X) - \mathrm{H}(X \mid Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{XY}(x,y) \log_2 \left(\frac{p_{XY}(x,y)}{p_X(x) p_Y(y)} \right)$$

which is a non-parametric measure for the mutual dependence [26] or correlations [23] between two random variables. For example, the mutual information is zero when X and Y are independent and attains a maximum of $\mathrm{H}(X)$ when $X = Y$. The definition can also be expressed as [23]

$$\mathrm{I}(X;Y) = d_{KL}(p_{XY} \| p_X p_Y)$$

with p_{XY} as the joint and p_X and p_Y as the marginal distributions of X and Y. Unlike KL divergence, mutual information is symmetric.

Another interpretation is it measures the reduction in uncertainty for X having learned Y [26]. In the study, X and Y correspond to the cell's environmental state and gene activity. And so, a model capable of high mutual information sends easily differentiable signals for the cell to know its environment and adapt.

To conclude, we present the **data processing inequality**, an intuitive but critical result. Suppose $X \to Y \to Z$[1] form a Markov chain, i.e. the conditional distribution of Z only depends on Y and that of Y only depends on X. Then

$$\mathrm{I}(X;Y) \geq \mathrm{I}(X;Z)$$

Informally, it captures how information degrades as it flows through more channels. When $Z = g(Y)$ for some transformation g, we precisely have a lower bound for the mutual information between X and Y given some processing of Y has occurred.

2.3.2 Transcription Signalling

Some traits of the transcription process have been described in past experiments. Finally, we lay out a few results that can serve as points for analysis in this study.

2.3.2.1 Transcriptional Bursting

Given the discontinuities of gene activity, the amount of RNA copies resulting from transcription in single cells has been characterised as exhibiting bursts or pulses on the time scale of minutes [27]. A simple yet capable model for this behaviour is one with possibly many gene activity states, some of which act as refractory states [11]. Models with many states can capture increasingly complex bursts. However, with more refractory states, the channel capacity, the maximum rate at which information can be transmitted, generally decreases [28]. Therefore, sufficient complexity is mostly context-dependent and requires multiple tuning steps to fit with data.

[1] This overloaded notation is not to be confused with chemical rates equations.

2.3.2.2 Transcription Factor Signals

In the context of stress adaptation in budding yeast, TFs have quantifiable differences. Some TFs have more variable nuclear concentrations than others leading to noisier transcription. Remarkably, TFs can also encode stress with varying specificities. For example, TFs **msn2** and **msn4** encode for general environmental stress in yeast, and TFs **Mig1** and **Yap1** encode for carbon and oxidative stresses, respectively [12]. Some can also discriminate against various stress intensities and use the time series of their traces in different ways.

Meanwhile, from an information-theoretic point of view, we can quantitatively establish the relationship between the true underlying transcription mechanism and the TFs. Suppose we have a model that depends on n independent TF signals $F_1, \ldots, F_n$, say, through some function g. By the Data-Processing inequality [25], we have

$$\mathrm{I}(X; g(F_1, \ldots, F_n)) \leq \mathrm{I}(X; F_1, \ldots, F_n) \leq \sum_{i=1}^{n} \mathrm{I}(X; F_i)$$

The sum of the mutual information for each TF and the environment is thus an upper bound for any such model. Despite this, signalling pathways are known to have redundancies to aid in noise suppression, and some TFs may potentially share some bits in common [12]. Hence, in addition to suitable complexity, a good model must be able to extract unique bits of information from the signals.

Chapter 3

Background

This chapter outlines existing methods from multiple disciplines. We assess their suitability to meet the study's goals and document current research related to ours. While the following chapter may appear to be verbose, it backs the many decisions concerning the study. Every concept below has been tested or at the very least considered at some point.

We first summarise approaches for simulating biochemical systems expressed as ordinary differential equations. Next, we cover recent practices for estimating mutual information from time series data. Finally, we discuss evolutionary algorithms for black-box optimisation and prompt the use of metrics and notions of similarity.

3.1 Stochastic Simulation of Biochemical Systems

The behaviour of a biochemical system can be written as a system of ordinary differential equations, typically known as *reaction-rate equations*. However, deterministic solutions may be difficult to compute in large systems. Furthermore, even if they exist, they fail to capture the stochasticity of biological phenomena. A technique often employed is to define a stochastic process by treating the ODE system as a master equation. Below, we present methods for simulating these stochastic biochemical systems.

3.1.1 Gillespie Algorithm

Gillespie [29] first presented a way to simulate coupled chemical reactions, producing an exact trajectory for a population of molecules. It is exact in that it considers the entire system regardless of its analytical solvability. The method is a Monte Carlo procedure that selectively fires reactions in between exponentially distributed time intervals. A pseudocode is given in Alg. 1. It and similar classes of algorithms are often known as exact **stochastic simulation algorithms** (**SSAs**).

A familiar analogy (see 2.2.2.1) can summarise the algorithm as follows: at the start, exponentially distributed alarm clocks are set, each corresponding to a chemical reaction and whose rate, also called its *propensity*, depend on an expression from the master equation. The time before the subsequent reaction is thus the minimum of the exponential random variables. This is also exponentially distributed, whose rate is the sum of propensities. The reaction attached to the first alarm clock that goes off is assumed to fire, and the procedure is repeated.

Algorithm 1: Gillespie Algorithm (Next Reaction Method) [30]

Input: initial state: $\mathbf{x_0}$, simulation time: $\mathbf{T}$

$\mathbf{x} \leftarrow \mathbf{x}_0$; $t \leftarrow 0$
while $t < T$ **do**
  $a_0(\mathbf{x}) \leftarrow \sum_{i=1}^{M} a_i(\mathbf{x})$ ▷ Calculate sum of propensities
  $\tau \sim \mathrm{Exp}(a_0(\mathbf{x}))$ ▷ Sample time until next reaction

  $u \sim \mathrm{U}(0,1)$ ▷ Sample uniform random variable
  Find $\mu \in [1,\ldots,M]$ where ▷ Choose reaction to fire
    $\sum_{i=1}^{\mu-1} a_i(\mathbf{x}) < u \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{\mu} a_i(\mathbf{x})$

  $\mathbf{x} \leftarrow \mathbf{x} + \nu_\mu$; $t \leftarrow t + \tau$ ▷ Update state and time
end while

While it is exact in trajectory, its computational complexity and non-deterministic simulation times may deter some use cases. More efficient ways of simulating the system have been proposed, including the Extrande algorithm [31]. However, a different strategy can be pursued by leaping through the simulation in constant intervals.

3.1.2 Poisson Tau-Leaping

In an attempt to address the *stiffness* of systems—the existence of multiple time scales, e.g. having fast and slow variables—a tau-leaping approximation was introduced [32]. Instead of sampling exponentially distributed times between reactions firing, one can sample the number of reactions that could have fired within a fixed interval from a Poisson distribution. Poisson tau-leaping incorporates this technique to produce quicker and, in most cases, a predictable number of time steps.

Nevertheless, it tends to be unreliable for smaller populations of molecules. For example, suppose we have two X molecules and some reaction R_X with molecule X as a reactant. A possibility is to randomly sample three instances of reaction R_X firing within a time step. This is impossible because insufficient reactants lead to a negative population.

Various efforts have been presented to overcome this, including leap rejection and falling back to an exact SSA [33], sampling from a Binomial distribution [34], and adaptively shrinking the step sizes [35]. While it is highly efficient for large populations, its main shortcoming is dealing with small populations, making it unsuitable for modelling binary or a small number of gene states.

3.1.3 Matrix Exponential Solution

Given a biochemical system whose population sizes for each molecule are decomposable into a finite state CTMC, a simple iterative application of its transition matrix allows a similar finite-step procedure. Rather than sample how many reactions go off in a given interval, one can sample the next state—a snapshot of the system—from the chain's transition matrix, given by the matrix exponential solution to the one-step chemical master equation

$$\mathbf{P}_t = e^{t\mathbf{Q}}$$

The procedure is reminiscent of methods using a discrete-time Markov chain (DTMC) to represent a CTMC. We note its distinction, however. One approach simulates the embedded

jump chain with exponentially distributed times between jumps. This is equivalent to Gillespie's algorithm. Another method called *uniformisation* incorporates holding times at each state into a DTMC given by

$$\mathbf{P} = \mathbf{I} + \frac{1}{q}\mathbf{Q}$$

where $q = \max\{q_{ii}\}_{i \in E}$, i.e. the maximum of the generator's diagonal. The number of state changes is sampled from a Poisson distribution and uniformly distributed across a fixed time interval $[0, T]$ [36]. These methods produce trajectories for the CTMC, whereas the proposed method produces snapshots of the CTMC's trajectories. That is, jumps could have occurred between two snapshots, but this becomes unlikely given a carefully chosen time step with respect to the holding times. Despite being inexact, a primary advantage is having constant time steps and thus vectorisation opportunities and allowing a natural approximation for non-homogeneous transition rates.

This method becomes impractical in the combinatorial explosion of population sizes and unusable in most settings where an unbounded number of molecules contribute to an infinite number of states. Transition probabilities within deeply coupled systems are not trivial either. A finite number of gene states does not suffer from these, making the method suitable.

3.2 Mutual Information Estimation

With trajectories of the gene's activity for each environment, the mutual information between the gene and the cell's environmental identity can be estimated. This quantity measures the distinctive quality of the gene's signalling responses under different stresses, indicative of the cell's decision-making capabilities. We highlight two recent methods for estimating mutual information from intracellular time series data.

3.2.1 Decoding-Based Estimation

A decoding-based approach first labels the trajectories of the gene's activity according to its environment. Next, a machine-learning classifier is fit on a portion of the data, and its performance is evaluated on a held-out data set [12]. This produces a confusion matrix

$$M = \left(\mathrm{P}\left(\hat{Y} = y_j \mid Y = y_i\right)\right)_{i,j \in E}$$

where Y and $\hat{Y}$ are the true and predicted environment labels respectively. The marginal probabilities can then be calculated to compute the mutual information. This yields a lower bound estimate given the information loss in training the classifier [12, 37], as the data-processing inequality requires. A highlight of this method is its flexibility with the choice of a classifier algorithm, a few of which have been previously tested.

Support vector machines (SVM) with radial basis functions have been shown to produce robust estimates regardless of the number of samples or dimensionality of the data [38], i.e. the length of the time series. However, when linear principal components are used to train the classifier under a different basis, it is unable to fully discriminate against dynamical patterns such as oscillations and random permutations [39]. Neural networks capture far more complex abstractions and have also been demonstrated to be comparable and better at times than SVM when a large number of samples are available [38]. Thus, the choice of a

classifier primarily depends on the size and complexity of the data as well as training and prediction speeds, among many others.

3.2.2 Stochastic Model-Based Estimation

A recent study by Tang et. al [37] proposed the use of Hidden Markov models (HMMs) and time-inhomogeneous DTMCs to learn the dynamics of a biochemical signalling molecule, i.e. a TF, and thus quantify the information accumulation within their signals. They propose a dynamic mutual information (dMI) measure capable of taking advantage of the entire time series, whereas MI estimates from other approaches saturate over time. It further addresses the insensitivity of other approaches to random permutations in the time series.

The resulting Markov chain structures are supposed to capture the activity of a single TF for a specific type of stress. We remark that these mainly serve as a stepping stone for the estimation of MI rather than the inference of TF properties, particularly given their large sizes—a 64-state model with 32 emission states, for example, was found optimal for the HMM. Moreover, training a HMM is rather computationally expensive compared to some classifier choices under the decoding-based method. They note a computational time on the order of 10 hours on a personal computer. While this may be acceptable for a limited number of ensembles, the specific goals of the study—one which involves the evaluation of over ten thousand different trajectory ensembles for each of multiple TFs—require otherwise.

3.3 Evolutionary Algorithms

Whereas differentiable functions can be optimised through gradient descent, functions with no direct access to their gradients require black-box optimisation techniques. Evolutionary algorithms (EA) are examples of such methods that mimic natural evolution to search for optimised solutions for black-box functions with a potentially vast parameter space. It is characterised by evolving a population of individuals, each corresponding to a point in the space [40].

We first outline genetic algorithms, a subclass of EAs, and further their use towards a multi-objective aim involving the diversity of individuals through novelty search.

3.3.1 Genetic Algorithms

The rationale behind **genetic algorithms** (**GA**) is to portray individuals based on their set of genes—their genome—and evolve the population through a Darwinian natural selection process. Holland [41] first proposed the idea with canonical representations of individuals taking the form of binary strings of fixed length [40], akin to genes. Like other EAs, individuals are subjected to randomised processes in the **selection**, **mutation**, and **crossover** operators.

Recently, the line between GAs and other EA subclasses, notably evolutionary strategies, has blurred with the introduction of far more complex genome structures. However, the focal point of GAs has ultimately remained the same, and for the rest of the paper, we shall address algorithms outlined here as GAs despite their non-canonical genomes.

An overview of the genetic algorithm process is as follows: a population of individuals is first evaluated based on their fitness, a possibly black-box measure of their quality. At this stage, elitism is a commonly employed technique to ensure the best fitness monotonically increases

per generation. Under elitism, a percentage of the fittest individuals are guaranteed a spot for the next generation [42]. Then, a set of individuals are selected as parents. Each pair of parents have their genes crossovered to yield two offspring. Offspring are then mutated by chance and added to the pool of individuals for the next generation. The process continues until the maximum number of generations is reached.

Undeniably, genetic operators heavily influence the evolutionary search. We outline their specifics and relevant developments.

3.3.1.1 Mutation

Mutations act on an individual's genome by adding slight variations, typically in a probabilistic manner. For example, bits can be randomly flipped in canonical binary strings [40], and Gaussian noise can be added to real-valued vectors [43].

The mutation rate controls a delicate balance between the search ground covered and the speed of convergence. With a low mutation rate, the search is not as incentivised to explore nearby solutions. With a high mutation rate, the search is more likely to escape local optima but at the cost of slower convergence. As the optimal rates vary per problem, adaptive mutation rates have been considered to avoid trial and error tuning [44]. More than one mutation operator may be used at a time, which can also be adaptive [45].

3.3.1.2 Crossover

A crossover, also known as a recombination, is the combination of features from two different individuals, forming two offspring individuals containing genes from both parents. A well-known example is a *uniform crossover*. Given two linear (one-dimensional) genomes of the same length, a random crossover point is uniformly sampled between the start and end of the genome. The genomes are split in half at these points, and the left cut of one parent is combined with the right cut of the other to form an offspring. An extension of this idea is selecting k points and alternating which parent an offspring gets its genes from. This is called an k-point uniform crossover [46].

When individual genomes take the form of graphs, a uniform crossover may still be performed on their adjacency matrices [47, 48]. However, this act of "flattening" the genome loses inherent graph properties such as connectivity. As such, a linear crossover does not guarantee feasibility, say, in the connectivity sense, in offspring. While penalising or discarding unfeasible individuals is possible, it is suggested that designing constraint-preserving crossover operators leads to a more efficient and less risky search [49]. Existing non-linear graph crossovers primarily split graphs into subgraphs and recombine pairs of subgraphs from different parents [50, 51, 52].

3.3.1.3 Selection

When selecting parents for crossover, fitter individuals are usually granted some advantage in being selected. The tendency to do so is known as *selective pressure*. With high selective pressure, a fit non-optimal individual may skew the search into having an identical non-optimal genotype. This decreases the genetic diversity, and the search prematurely converges [53]. Two examples of selection schemes are discussed.

Roulette wheel selection chooses individuals proportional to their fitness values. In a

population of N, each individual a_i has a probability of being selected, given by

$$\mathrm{P}(a_i) = \frac{f(a_i)}{\sum_{j=1}^{N} f(a_j)}$$

where f is the fitness function [53]. The roulette wheel is spun n times, choosing n individuals. In this scheme, the magnitude of the difference between fitnesses influences the likelihood of selection. As such, nearly identical fitness values make it difficult to move towards better individuals. Moreover, while it promotes diversity by giving each individual a chance of getting selected, an especially fit individual at the start may lead to premature convergence [54]. Holland [41] originally employs a similar method.

On the other hand, k-**tournament selection** first selects k individuals at random. They then compete in a tournament, and the individual with the highest fitness is chosen [54]. Unlike the roulette wheel, the specific fitness values do not affect the selection. Its efficiency and parallelisability have also made it a popular method with binary ($k = 2$) and slightly larger tournaments as common choices [53]. Furthermore, the choice for k allows a controllable parameter for selective pressure.

3.3.1.4 Multi-Objective Search

Classical GAs aim to optimise individuals based on a single fitness value. In practice, it may be desirable to quantify an individual's fitness in more than one way. For example, a product may be evaluated based on its manufacturing cost and structural durability. Multi-objective searches use the notion of **Pareto dominance** to address this.

Suppose the target function to maximise, $\mathbf{f}$, has n components as follows

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$$

for some vector $\mathbf{x}$ in the space of solutions $\mathcal{U}$. For $\mathbf{x}_u, \mathbf{x}_v \in \mathcal{U}$, let $\mathbf{u} = \mathbf{f}(\mathbf{x}_u)$ and $\mathbf{v} = \mathbf{f}(\mathbf{x}_v)$. We say $\mathbf{x}_v$ *dominates* $\mathbf{x}_u$ if

1. All its entries are greater or equal: $\forall i \in \{1, \ldots, n\},\ v_i \geq u_i$.
2. At least one of its entries is greater: $\exists i \in \{1, \ldots, n\} \mid v_i > u_i$.

A decision vector $\mathbf{x}_u$ is defined to be *Pareto-optimal* if there is no $\mathbf{x}_v \in \mathcal{U}$ which dominates it [55]. The set of all Pareto-optimal solutions is called the *Pareto front*. It is certainly possible that a solution neither dominates nor is dominated.

A notable development is the **non-dominated sorting genetic algorithm** (**NSGA**), which took a Pareto-based approach and was later adapted to **NSGA-II** to incorporate elitism [56]. It sorts individuals according to their front number, i.e. the number of individuals it is dominated by, and uses this over traditional fitness for elitism and selection. However, it is found that Pareto fronts usually lack diversity. A crowding distance measure was suggested to alleviate such concerns [57]. The next subsection examines a diversity-focused search that is eventually paired with NSGA-II.

3.3.2 Novelty Search

Lehman and Stanley [58] highlight the deceptive landscape of objective (fitness) functions and their affinity towards local optima as main proponents of a search for *novelty*. They argue

that objective functions may not incentivise reaching *stepping stones*, potentially sub-optimal points in the search space that can lead to the overarching objective. Their initial proposal is driving the evolutionary search without an objective function but simply with novelty.

A measure for the novelty of an individual $\mathbf{x}$ is defined to be the average distance to its k nearest neighbours,

$$\rho(\mathbf{x}) = \frac{1}{k}\sum_{i=0}^{k} d(\mathbf{x}, \mu_i)$$

where d is a distance measure, also called the *novelty metric*, and μ_i is the ith nearest neighbour of x under the measure d. An archive of past individuals with novel behaviours is maintained throughout, and the distance calculations are done with respect to both the archive and the entire population. Individuals are added to the archive when their novelty score exceeds some threshold. This incentive for diverging behaviour, however, potentially makes convergence difficult.

3.3.2.1 Local Competition

A follow-up to their novelty search, Lehman and Stanley [59] introduce fitness through competition for a multiobjective search involving fitness and novelty, thereby rewarding good performance and diverse morphologies. It uses an NSGA-II scheme without needing an explicit diversity-preserving measure within Pareto fronts, as the novelty objective is assumed to capture genotypic diversity already.

The local fitness score of an individual $\mathbf{x}$ is defined to be the number of its k nearest neighbours for which it outperforms,

$$f_{local}(\mathbf{x}) = |\{\mu_i \in \text{neighbours}(\mathbf{x}) \mid f(\mathbf{x}) > f(\mu_i)\}|$$

When pitted against fitness-only, novelty-only, and novelty with global competition schemes, it finds comparable maximum absolute performance. Further, it outperforms the others in terms of its exploration of niche behaviours. Methods with this attribute of finding fit and diverse individuals are summarised under Quality-Diversity algorithms [60].

3.3.2.2 Archival Management Strategies

In traditional novelty search, the archive serves as a reference for novelty calculations of the current population. An intuitive extension sees the archive as the end result of the search and treats its maintenance as building a repertoire of behaviours [61]. A novelty search takes place, but when an individual is found to be better than a nearly identical individual in the archive, it replaces the archived individual.

Archives can primarily be classified as structured or unstructured, depending on an individual's genome. For example, individuals with a vector representation, albeit high-dimensional, can be allocated on a discrete grid where cells correspond to some range of parameter values [62]. A single individual occupies each cell, and the search drives these individuals to be fitter over time through replacement. This structure allows for an interpretable collection of behaviours. In the unstructured case, where only the distances between individuals are available, care must be taken to arrive at an analogous uniform collection.

In an unstructured archive, this technique can lead to an uneven density of the collection and *erosion* of its border. A maximal density parameter, i.e. a fixed novelty threshold, and a

modified Pareto dominance scheme for replacing archived solutions have been sufficient to resolve these issues [61]. The difference is as follows: given a novelty threshold l, an individual is archived if its distance to its nearest neighbour is less than l, its distance to its second nearest neighbour is greater than l, and within some error ϵ, the individual dominates its nearest neighbour.

3.4 Metrics and Similarity Instruments

Deciding the similarity between two objects is a task common to many domains. In particular, novelty search necessitates a well-defined distance measure between individuals and their efficient calculation. This section summarises the use of metrics for search and relevant similarity measures.

3.4.1 Similarity Search in Metric Spaces

Given a set of objects $\mathbb{X}$ and any $x, y, z \in \mathbb{X}$, a function $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ is said to be a **distance metric** if it satisfies the following properties:

1. Reflexivity: $d(x, x) = 0$
2. Positivity: $d(x, y) \geq 0$
3. Symmetry: $d(x, y) = d(y, x)$
4. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

If d is a metric, then the pair $(\mathbb{X}, d)$ is called a **metric space**. While these seem to be strict well-behaved mathematical objects, they appear in many practical applications such as database retrieval and video compression, most notably for similarity searches [63].

The data set of objects $\mathbb{U} \subseteq \mathbb{X}$ considered within the metric space has an important quality in its **intrinsic dimensionality**. This concept refers to the minimum number of parameters sufficient to express or describe the data [64]. For example, the set of all three-dimensional points lying on a specific plane has an intrinsic dimensionality of two despite its three-dimensional representation. In fact, there are ways to estimate this quantity directly from the data set [64, 65].

The curse of dimensionality has large effects on a metric space. As an example, consider an arbitrary d-dimensional point lying within a unit hypercube. It can be shown that the volume of a unit hypersphere divided by that of a unit hypercube goes to zero as the dimension d tends to infinity [66]. That is to say, much of high-dimensional space is far from the origin. Now, framing any point as the origin for reference precisely explains why the average distance between a point and its nearest neighbours increases with dimension. For relatively low dimensions, however, metric spaces still offer desirable properties.

Space-partitioning data structures go hand in hand with metric spaces to allow efficient queries for proximity and nearest neighbour (NN) searches, often by constructing a tree where each subtree partitions the search space. For arbitrary metric spaces, several examples include vantage point trees, also called metric trees [67], M-trees [68], and cover trees [69]. These methods primarily rely on the triangle inequality to prune the search and are effective for low-dimensional data. Whereas a brute-force NN search requires $O(n)$ complexity, a cover tree, for example, requires $O(c^{12} \log n)$ in the worst case, where c is dubbed the bounded expansion constant. To give some context, in a uniform data set $\mathbb{U}$ with dimension

d, $c \sim 2^d$ [69]. In high dimensions, the distances between points become nearly identical and the pruning condition is almost never met. Therefore, these methods require a careful judgement of the data's intrinsic dimensionality.

3.4.2 Statistical Distances

Multiple measures exist that quantify the dissimilarity between two probability distributions. Below, we highlight two of significance.

3.4.2.1 Jensen-Shannon Divergence

Suppose we have two discrete probability distributions $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$. Recall that the KL divergence between P and Q is

$$d_{KL}(P\|Q) = \sum_{i=1}^{n} p_i \log_2 \left(\frac{p_i}{q_i}\right)$$

which is a non-symmetric measure. The **Jensen-Shannon divergence** offers a symmetric, bounded, and well-defined extension, as given by

$$\begin{aligned} d_{JS}(P\|Q) &= \frac{1}{2} d_{KL}\left(P\|R\right) + \frac{1}{2} d_{KL}\left(Q\|R\right) \\ &= \mathrm{H}(R) - \frac{1}{2}\left[\mathrm{H}(P) + \mathrm{H}(Q)\right] \end{aligned}$$

where $R = (P+Q)/2$ [70]. Like KL divergence, it has some information-theoretic interpretations [71]. Furthermore, it has been shown to be the square of a distance metric [70], making it suitable for metric space construction.

3.4.2.2 Wasserstein Distance

Suppose we have two probability distributions μ and ν on a metric space $\mathbb{X}$. The p-**Wasserstein distance** between μ and ν is analytically given as

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{X}\times\mathbb{X}} d(x,y)^p \, \mathrm{d}\gamma(x,y)\right)^{1/p}$$

where $\Gamma(\mu, \nu)$ is the set of probability measures γ on $\mathbb{X} \times \mathbb{X}$ with $\gamma(A \times \mathbb{X}) = \mu(A)$ and $\gamma(\mathbb{X} \times B) = \nu(B)$ and d is often the Euclidean distance [72]. Among many other names, it is also known as the *earth mover's distance* with its ties to optimal transport theory. One distribution can be interpreted as the mass of dirt spread across some space and the other as a set of holes within the same space. This measures the minimum work required to fill the holes with dirt, where the cost is proportional to the amount of dirt and the ground distance covered [73].

A useful result is that W_p is a distance metric and persists to be a metric when $(\mathbb{X}, d)$ is a metric space [72]. Combining this together with a scheme for graph isomorphism tests, Togninalli et. al [74] proposed a distance metric between labelled graphs—the *Wasserstein Weisfeiler-Lehman distance*. In it, they consider the $p = 1$ discrete case, where the distributions are given by two vectors $X \in \mathbb{R}^{n\times m}$ and $X' \in \mathbb{R}^{n'\times m}$ resulting in

$$W_1(X, X') = \min_{P \in \Gamma(X,X')} \langle P, D \rangle$$

where D is the pairwise distance matrix between all elements in X and X'; $P \in \Gamma$ is the transport matrix, often assumed to be a uniform distribution; and $\langle \cdot, \cdot \rangle$ is the dot product. First, the Weisfeiler-Lehman embedding scheme is iteratively performed m-times to produce feature vectors X and X' from graphs G and G'. For discrete labels, we have

$$X = \begin{pmatrix} l^0(v_1) & l^1(v_1) & \dots & l^m(v_1) \\ \vdots & \vdots & \ddots & \vdots \\ l^0(v_{|G|}) & l^1(v_{|G|}) & \dots & l^m(v_{|G|}) \end{pmatrix}, \quad l^h(v) = \begin{cases} l(v) & h = 0 \\ \text{hash}(l^{h-1}(v), \mathcal{N}^{h-1}(v)) & h \geq 1 \end{cases}$$

where $l(v)$ is the original label of vertex v and $\mathcal{N}^h(v)$ is the set of labels at iteration h of vertices neighbouring v. Ideally, a perfect hash is used. Then, the Hamming distance is computed between each graph embedding X and X', yielding the matrix M. Finally, a network simplex method is employed to calculate the expression [74].

3.5 Related Work

The study again aims to infer interpretable transcription models for yeast cells that maximise the mutual information between a cell's environmental identity and gene activity. In doing so, it naturally aims to develop an efficient methodology for such tasks. In this section, we briefly highlight similar works and their significance.

Granados et. al [12] first explored how extracellular information may be organised inside a yeast cell. They consider multiple TFs and measure their nuclear concentration after inducing different types and levels of stress across trials. The mutual information between the TF traces and the environment, together with discrete stress levels, were quantified, and analysis focused on the encoding specificity and redundancy of TFs. The study builds upon their work by proposing concrete transcription regulation mechanisms with their measurements as ground truth.

A precursor to this study is the work of Bobrowski [13], which considered a finite set of small (2-4 state) transcription models with a similar goal of optimising their mutual information. Each model took the form of a DTMC and was given a fixed structure. A pipeline was established with the Extrande algorithm to simulate single trajectories and an SVM decoder for estimation. Particle swarm was used to optimise the reaction rates. However, due to the high complexity of exact SSA methods and the strong constraints involved with the model architectures, their method becomes intractable for larger, more general situations. The study improves upon their methodology by proposing a more efficient pipeline set-up and a genetic algorithm scheme that searches through an arbitrarily large solution space.

A number of studies have described types of promoter architectures with discrete activity states [75, 11]. A few, however, have derived architectures through systematic procedures from experimental findings. An example is the work of Neuert et. al [10], which derived a four-state transcription model for yeast cells undergoing osmotic stress. Their study first considers two- to five-state models and fits their time-varying probability distributions with their experimental mRNA distributions, allowing transition rates to depend on one TF. Then, after a cross-validation test indicates a four-state model's expressive yet non-overfitting choice, they demonstrate its generality by showing its ability to fit data when a different TF is used. As is typical with inference studies, a model's performance is compared against the true mRNA expression. This study, however, goes against this convention by maximising a mutual information measure instead.

Chapter 4

Transcriptional Architectures

This chapter illustrates how the study models gene activation. We first define promoter models together with their constraints. Lastly, we put forward a method for randomly generating promoter models given a fixed number of states.

4.1 Promoter Models

A promoter model is framed as a time-inhomogenous continuous-time Markov chain (CTMC) whose states describe the amount (or likelihood) of activity within the promoter region. In the most general sense, each state has an associated weight, all of which sum to one, e.g. an inactive state has weight zero. Note the system is decoupled from mRNA and reduced to gene activity, as will be justified in the next section.

We define models by an *activity array* $\mathbf{A}$ and a generator matrix $\mathbf{Q}$ whose entries correspond to the exponential rate parameters associated with jumping between states. These rates are considered to either be constant or linearly dependent on the concentration of a TF. As the concentration changes with time, we express the generator as a function of time.

E.g. the generator and activity array below defines the model in Fig. 4.1.

$$\mathbf{Q}(t) = \begin{bmatrix} q_{11} & 0.459 & 0 \\ 4.244{\cdot}\mathrm{TF}_1(t) & q_{22} & 0.131{\cdot}\mathrm{TF}_2(t) \\ 0 & 0.055 & q_{33} \end{bmatrix}$$

$$\mathbf{A} = [0.66,\ 0.00,\ 0.34]$$

where $q_{ii} = -\sum_{j \neq i} q_{ij}$ and $\mathrm{TF}_X(t)$ is the concentration of a TF X at time t.

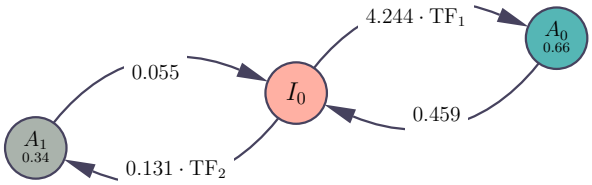

Figure 4.1: A three-state promoter model with one inactive and two distinct active states.

We further explore three paradigms of gene activity with increasing generality and abbreviate them for conciseness. The first, **one active state** (**OAS**), is the simplest yet most widely accepted belief in current research. In OAS, a single state has a weight of one, and the rest have a weight of zero. The second, **multiple active states** (**MAS**), is less strict, allowing more complex dynamics. MAS deals with multiple discrete, i.e. indistinguishable, active states. The third, **spectrum of weighted activities** (**SWA**), is yet another viewpoint held by studies

to explain more complicated mRNA traces. SWA is the largest superset allowing states to take on continuous, likely different, activity values.

From an information-theoretic point of view, SWA is more capable of producing distinctive activity traces. However, this comes at the cost of a significantly larger solution space. These classes represent differing complexities involved in modelling transcription bursts.

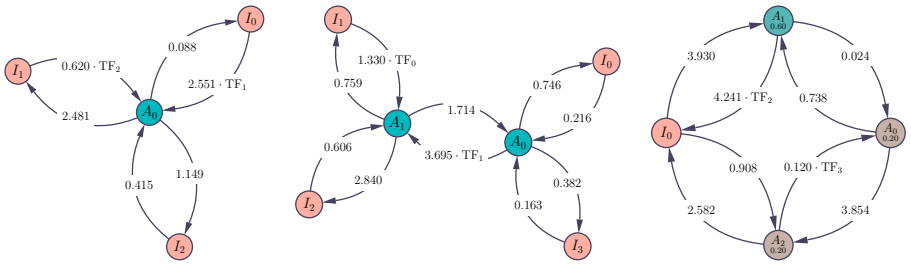

Figure 4.2: Example models under paradigms: (Left) OAS, (Middle) MAS, and (Right) SWA.

4.1.1 Model Constraints

Irrespective of their classes, promoter models are also assumed to follow a main set of constraints given as follows:

1. **Connectedness**: every state must be reachable from every other state, i.e. the chain is irreducible. This is a natural simplification to avoid isolated components.
2. **Reversibility of reactions**: if there is an edge connecting state i to j, there must be an edge connecting state j to i. This is biologically motivated, given the binding mechanisms of TFs.
3. **At least one active state**: at least one state with non-zero weight must exist. For ease, we impose this constraint on the first state. This rids of trivial inactive trajectories.

We call a model *infeasible* if it fails to meet any of the constraints.

4.2 Random Model Generation

Given a fixed number of states, we propose a way to construct an arbitrary model. Note that the reversibility constraint allows us to treat the model as an undirected graph.

First, a uniform spanning tree is sampled via Wilson's loop-erased random walk [76] to construct an undirected skeleton graph. Next, each remaining unconnected edge is linked with some fixed probability, e.g. 0.5. This produces a connected, undirected graph.

With the connections finalised, the skeleton is mapped back to a directed graph with twice the number of edges. Then, each edge is assigned a random rate function. We allow them to be constant or linearly dependent on a randomly assigned TF. The rates are then uniformly sampled between $[-2, 2]$ in the $\log_{10}$ space. This is due to the time scales of the data set.

Finally, states are assigned an activity. The first state is always set to be active to meet the last constraint. In MAS and SWA, succeeding states are labelled active with some fixed probability, e.g. 0.33. For OAS and MAS, active states are given a weight of one. For SWA,

they are assigned a random weight from a standard uniform distribution. The activities are then normalised to exact a sum of one. This is summarised in Alg. 2.

Algorithm 2: Random Model Generation

Input: no. of states: $\mathbf{N}$, prob. of forming an edge: `p_edge`, prob. of active state: `p_active`

`generator` $\leftarrow \mathbf{N} \times \mathbf{N}$ array of zeros
`activity` $\leftarrow \mathbf{N} \times 1$ array of zeros
`skeleton` $\leftarrow$ `sample_uniform_spanning_tree`$(\mathbf{N})$

for (i, j) in `skeleton.unconnected_edges`, with $i < j$ **do**
    $u \sim \mathrm{U}(0, 1)$
    if $u <$ `p_edge` **then**
        `skeleton`$[i, j] \leftarrow 1$
        `skeleton`$[j, i] \leftarrow 1$
    end if
end for

for (i, j) in `skeleton.connected_edges` **do**
    `generator`$[i, j] \leftarrow$ `get_random_rate_function`$()$
end for

if OAS **then**
    `activity`$[0] \leftarrow 1$
else if MAS **then**
    `activity`$[0] \leftarrow 1$
    for $n \leftarrow 1 \ldots \mathbf{N} - 1$ **do**
        $u \sim \mathrm{U}(0, 1)$
        if $u <$ `p_active` **then**
            `activity`$[n] \leftarrow 1$
        end if
    end for
else if SWA **then**
    `activity`$[0] \sim \mathrm{U}(0, 1)$
    for $n \leftarrow 1 \ldots \mathbf{N} - 1$ **do**
        `activity`$[n] \sim \mathrm{U}(0, 1)$
    end for
end if
`activity` $\leftarrow$ `activity`$/ \sum_{n=0}^{\mathbf{N}}$ `activity`$[n]$

Our method satisfies the constraints and allows tunable parameters for the sparsity and activity levels of models. For instance, shown in Fig. 4.3 are increasingly complex architectures.

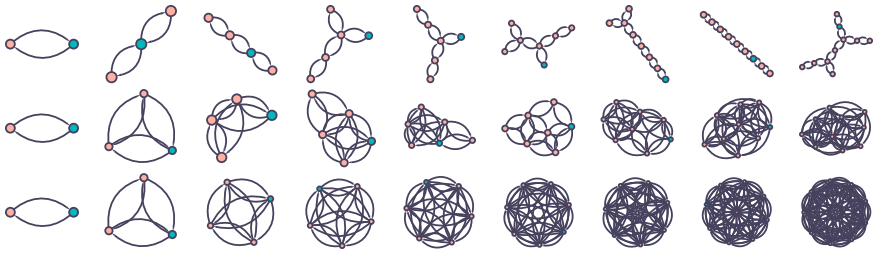

Figure 4.3: Silhouettes of randomly generated models. Columns indicate the number of states ranging from two to ten, and rows correspond to values $p_{edge} = 0, 0.5$, and 1.

Chapter 5

Evaluation Pipeline

This chapter introduces a pipeline for evaluating the performance of any promoter model. First, its trajectories under different environments are simulated to produce a collection of time series data for gene activity. The mutual information (MI) between its activity and the environment it encodes is then estimated under a decoding-based framework.

5.1 Promoter Activity Simulation

Given an experimental data set containing the nuclear concentration of individual TFs, the generator of a model $\mathbf{Q}(t)$ can be realised across the time stamps for which measurements were recorded. This yields a collection of instantaneous generators $\{\mathbf{Q}(t_0), \ldots, \mathbf{Q}(t_{N_t})\}$. We have multiple instances of these sets, as measurements are replicated across N_{cells} trials, typically over a hundred, and across N_{env} different environments. As stochastic simulations tend to be noisy, particularly for those with discrete activities, we further perform N_{reps} replicates. In total, we have $N_{env} \times N_{cells} \times N_{reps}$ trajectories to simulate from these generators. We later show the method can do so under the time it takes one SSA trajectory.

5.1.1 Simulation through Matrix Exponentials

Most interesting biochemical systems have coupled reactions involving many molecules. In the case of a transcription model, the only substances are the gene states and the mRNA it produces when active. However, the data processing inequality assumes a lower MI with mRNA traces compared to the gene's activity. For this reason, mRNA can be decoupled from the system, leaving a master equation solely for switching finite gene states. In this scenario, an exact SSA decomposes into a CTMC simulation. Given explicit generators for our CTMC model, we can take fixed step sizes through the simulation.

The basis of this method lies in the (one-step) chemical master equation and its solution

$$\mathbf{P}_t' = \mathbf{P}_t\mathbf{Q}, \quad \mathbf{P}_t = e^{t\mathbf{Q}}$$

where $\mathbf{P}_t$ is the transition matrix after an elapsed time t, and $\mathbf{Q}$ is the generator matrix of the chain. Given $\tau > 0$, the transition matrix at time $t + \tau$ can be given as

$$\mathbf{P}_{t+\tau} = \mathbf{P}_t\mathbf{P}_\tau = \mathbf{P}_t e^{\tau\mathbf{Q}}$$

However, as we are dealing with a time-inhomogeneous chain, the above results generally do not hold. This is where an approximation is taken. Provided a carefully chosen time step τ, we may assume the chain is time-homogeneous and derive the expression

$$\mathbf{P}_{t+\tau} = \mathbf{P}_t e^{\tau \mathbf{Q}(t+\tau)}$$

Note then that, at each time step t, a probability distribution for the states at $t + \tau$ can be computed through $\mathbf{x}_t \mathbf{P}_{t+\tau}$ where $\mathbf{x}_t$ is a binary vector representing the chain's state at time t. The next state is thus chosen based on this distribution. Beginning with the chain in one state, we can efficiently step through the simulation. A main improvement over exact SSA is its parallelisability and fixed simulation time. This is also helpful as experimental measurements are taken in constant time intervals.

5.1.2 Batching of Cells and Replicates

Matrix exponentials are first pre-computed by batching them together for each environment, cell sample, and time step. After conducting tests, a Pade approximation [77] was chosen as Taylor series expansions often led to a non-stochastic $\mathbf{P}_t$ and would only converge past 20 terms. As calculations overflow for large parameters, the generator is first scaled by some 2^k and subsequently used to calculate $\exp(\tau \mathbf{Q}) = \left(\exp(\frac{\tau}{2^k}\mathbf{Q})\right)^{2^k}$. The choice of powers of two allows more straightforward matrix exponents.

After rearranging the axes as appropriate, this leads to a tensor $\mathbf{G}$ with

$$\mathbf{G} \in \mathbb{R}^{N_t \times N_{env} \times N_{cells} \times N_{states} \times N_{states}}$$

The simulation can thus be carried out. A state $\mathbf{x}_{t_0}$ is first uniformly chosen. Then, at each time step, a probability distribution for the next state is calculated, and a state is randomly selected. Upon choosing a state i at time t, we aim to calculate the probability distribution of the next state, i.e.

$$[\mathbf{P}_{t+\tau}]_i = \left[\mathbf{P}_t e^{\tau \mathbf{Q}(t+\tau)}\right]_i = [\mathbf{P}_t]_i e^{\tau \mathbf{Q}(t+\tau)}$$

We assign $[\mathbf{P}_t]_i = \mathbf{x}_t$, which is a vector of zeros except at the ith entry. Hence the iterative procedure can be continued until the simulation ends. Notice also that if we never realise the chosen state, that is, keep $[\mathbf{P}_t]_i$ as a probability distribution, then this produces a time series of the state distribution of the chain. This fact is used later on.

To perform all these calculations efficiently, they are similarly batched. Denote by the tensor $\mathbf{D}[t]$ the batched probability distributions of the next state at time t, i.e. $[\mathbf{P}_t]_i$ if i was the previous state, with

$$\mathbf{D}[t] \in \mathbb{R}^{N_{env} \times N_{reps} \times N_{cells} \times N_{states}}$$

This can be used to compute another tensor of the same shape, $\mathbf{S}[t]$, the indicator of the current state. We do so by finding the cumulative sum of $\mathbf{D}[t]$ in the last axis, sampling a tensor $\mathbf{R}$ from a uniform distribution, and searching the maximum indices in $\mathbf{D}[t]$ for which the uniform numbers are less than its entries.

At each time point t, we thus calculate the product of the tensors

$$\mathbf{D}[t+\tau]_{ijkm} = \sum \mathbf{S}[t]_{ijkl} \mathbf{G}[t]_{ijlm}$$

and again solve for the state tensor. In doing so, we find a time series of the chosen states $\mathbf{S}$. This collectively simulates $N_{env} \times N_{cells} \times N_{reps}$ trajectories together.

5.1.3 Benchmarks

5.1.3.1 Time Scales

Our method relies on the value of τ for the time-homogeneity assumption to be reasonable. Unlike the Gillespie algorithm, the time steps of our method are fixed, and thus its simulation times are not affected when rates are large. And so, we briefly explore if there are any effects when rates are varied under our method. Consider, for example, a generator $\mathbf{Q}(t)$ and a choice for τ. Now, suppose we scale all the rate parameters by k. Then we assume

$$\mathbf{P}_{t+\tau} = \mathbf{P}_t e^{\tau k \mathbf{Q}(t+\tau)}$$

which is equivalent to a choice of $\tau' = k\tau$.

When $\tau \gg 1$, the average trajectory approaches a fixed value. To see this, let $\mathbf{1}$ be a vector of ones and consider the homogeneous case where $\mathbf{Q}_\delta = (\mathbf{P}_\delta - \mathbf{I})/\delta$. Then

$$e^{\tau \mathbf{Q}}\mathbf{1} = \lim_{\delta \to 0} e^{\tau \mathbf{Q}_\delta}\mathbf{1} = \lim_{\delta \to 0} \left(\mathbf{I}\mathbf{1} + \sum_{i=1}^{\infty} \tau \mathbf{Q} \frac{\delta^i}{i!}\mathbf{1} \right) = \mathbf{1}$$

and so the matrix exponential of the generator (of a stochastic semigroup) is stochastic. For our fixed generator $\mathbf{Q}(t+\tau)$, we also have that $e^{\tau \mathbf{Q}(t+\tau)}$ is stochastic. Recall that the original chain is irreducible (and finite) by our model constraints, and so entries in $e^{\tau \mathbf{Q}(t+\tau)}$ are non-zero. Consider the DTMC with a transition matrix equal to this matrix. It also inherits irreducibility and hence a unique stationary distribution exists. For k large enough, $[(e^{\tau \mathbf{Q}(t+\tau)})^k]_i$ is equivalent to its stationary distribution, and it becomes idempotent.

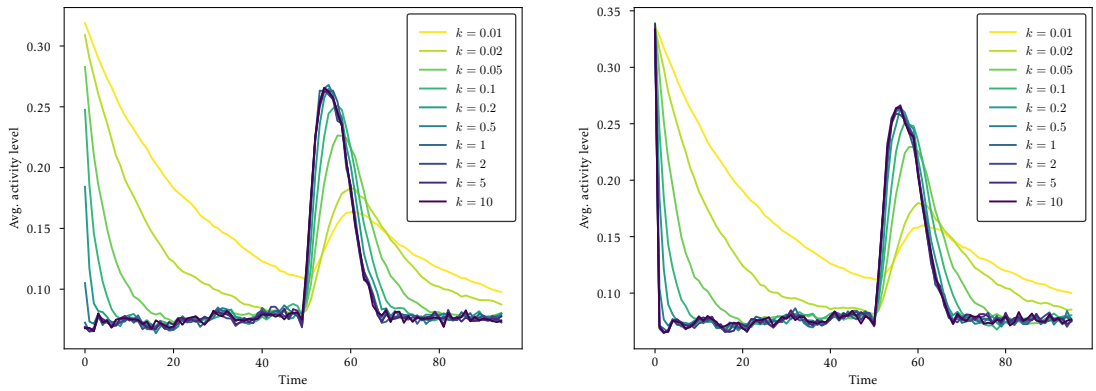

Figure 5.1: Average activity of a yeast cell under carbon stress using (Left) Gillespie algorithm and (Right) our approximation for values of $k = 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 5, 10$. Data used for the simulation is taken from Granados et. al [12]. Note that the initial state is uniformly chosen, so activity begins at 0.33. These trends are an average of 100 replicates.

Shown in Fig. 5.1 are simulations under the Gillespie algorithm and our method for the promoter model defined by the generator and activity array given below

$$\mathbf{Q}(t) = k \cdot \begin{pmatrix} -q_{11} & \frac{1}{2} & 1 \\ \mathrm{TF}_2(t) & -q_{22} & 1 \\ \frac{1}{2}\mathrm{TF}_3(t) & 1 & -q_{33} \end{pmatrix}, \quad \mathbf{A} = [1,\ 0,\ 0]$$

In both methods, smoothing occurs for small k and higher peaks form for large k.

5.1.3.2 Simulation Times

Using the scheme in Section 4.2, we randomly generate models and benchmark the average time required to produce their trajectories. As standard deviations for the Gillespie algorithm were found to be larger than the mean, the interquartile range is shown instead. The trendlines are given below in Fig. 5.2. While our method's execution times are not affected

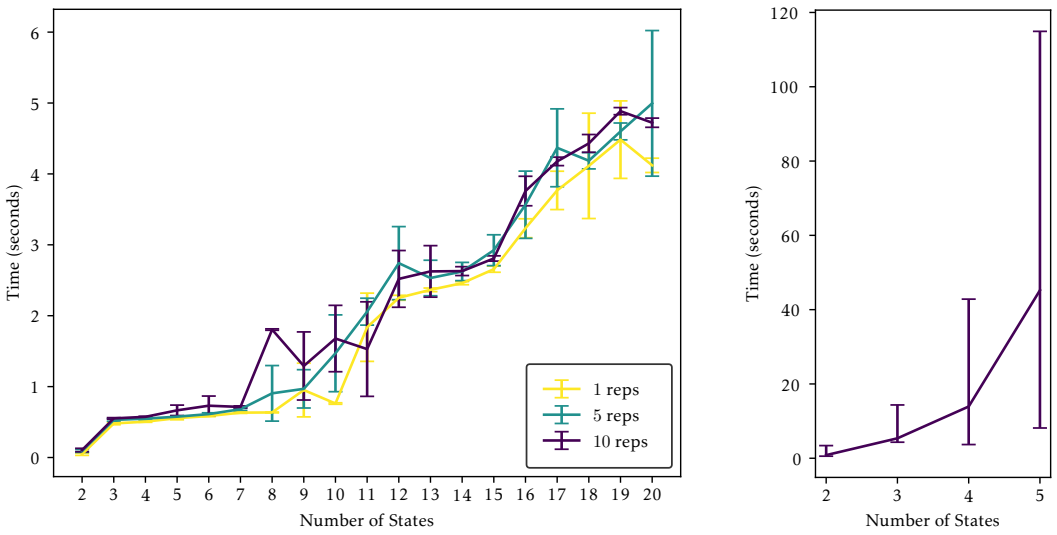

Figure 5.2: The average time taken under our method (left) and the median time for the Gillespie algorithm with one replicate (right) as the number of states increases. Despite increasing replicates, vectorisation within our method can keep simulation times within the same order of magnitude. Error bars are standard deviations on the left and interquartile ranges on the right. Each data point is an average across ten random models. TF data is also from Granados et. al [12]. Simulations were run on a machine with Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz.

by the transition rates, Gillespie simulations tend to have an explosion of reactions when the rates are high. Its duration increases with the number of states as more reactions can fire. Further, with more states, the proportion of high rates varies more, likely causing large variances in simulation times.

With more than five states, our method clearly outperforms the Gillespie algorithm in speed and consistency of times. To make a fair comparison, we may consider the case of four-state models, one of the more common promoter structures in studies. Note that we can increase the throughput of our method with more replicates. However, there is not much incentive past ten, which we later discuss in the next section.

A single simulation produces $\sim$400 trajectories for one replicate and $\sim$4000 trajectories for ten. With each run taking $\sim 0.5s$ for a four-state model, our method has a throughput of one trajectory per $\sim 1.25 \times 10^{-4}s$. With one replicate under the Gillespie algorithm, each run takes $\sim 20s$, amounting to a throughput of one trajectory per $5 \times 10^{-2}s$.

5.1.3.3 Memory Management

Matrices grow quadratically in size with the number of states. The section of code for which the memory consumption peaks is when matrix exponentials are pre-computed. Batching

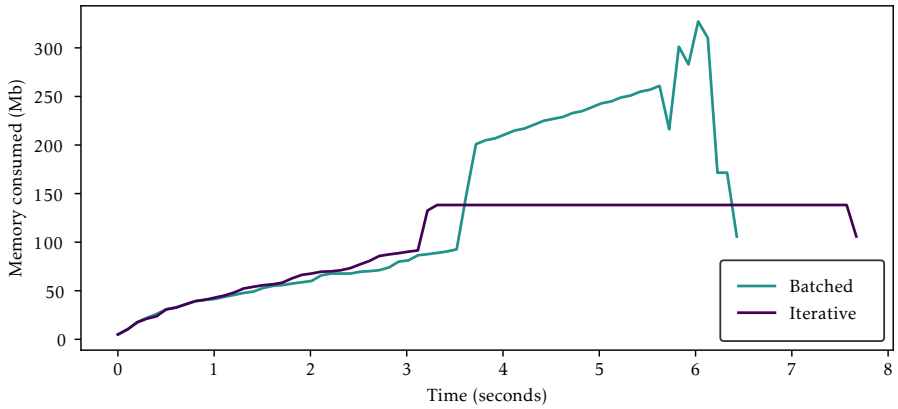

Figure 5.3: Memory trends when a program simulating a 15-state model is profiled. Simulations begin at around the $3.5s$ mark. Under a batched scheme, memory peaks twice as high compared to an iterative scheme. Note that the total memory used at every $10^{-1}s$ is sampled due to profiling, adding to the simulation times.

them in five-dimensional tensors allows for quicker times but at the cost of a larger memory footprint. Due to the constraints involved with total memory allowed within machines used for the study, we iteratively compute matrix exponentials for models with over 15 states. This reduces the memory consumption by more than half, as shown in Fig. 5.3.

5.2 Mutual Information Estimation

The next step of the pipeline is evaluating the mutual information from the collection of trajectories labelled under different environments. In the study, we work with a decoding-based approach.

5.2.1 Machine Learning Pipeline

Each trajectory corresponds to a cell being transferred from a rich to a stressful environment. We call the time at which this transfer takes place as the *origin*. First, we splice the trajectories M intervals before and after the origin. This produces two M-length cuts, one corresponding to the gene's status in a rich environment and the other in a stressful environment. Given we have N_{env} environments, each with N_{traj} trajectories. We will have N_{env} times as many rich cuts as any other environmental stress cuts. We down-sample by randomly choosing N_{traj} out of all the rich cuts to maintain an equal number of samples per environment. Each of the cuts is then labelled according to its environment.

Recall that we employ replicates in trajectories to capture a wider range of behaviour from the possibly noisy trajectories. When shuffling the data, we require that the replicates corresponding to the same cell sample are put together to ensure no data leakage leads to overestimates of MI. Fifteen per cent of the data is assigned as the validation set, whereas the rest is used for training and testing.

We then have a classification task to solve. In the case of a trivially poor model, all its trajectories may be inactive. However, certain classifiers are not able to handle inputs having

multiple labels. To deal with this, if the variance within the validation set is below a certain threshold, an MI of zero is assigned prematurely.

The general pipeline is then as follows: input vectors are standardised, their principal components are extracted, and these are fed to the classifier. A halving grid search is performed with the validation set to tune the classifier's hyperparameters under this pipeline. With the remaining 85% of the data, it is split 70-15 for training and testing across a number of bootstraps. For each bootstrap, we calculate the marginal probabilities from the confusion matrix and solve for the MI. The mean of which is the output.

5.2.2 Classifier Selection

The ideal classifier is fast in its tuning, training, and predictions and accurate in its assessments. We consider four different classifiers that may fall under this umbrella: **support vector machines** (**SVM**), **random forest** (**RF**), **decision trees** (**DT**), and **naive Bayes** (**NB**). We note that k-NN with dynamic time warping and neural networks were tested but were considerably slower, despite possibly providing better performance.

This part of the study is a balancing act. We aim to have the simulation and decoding times within the same order of magnitude for an accurate and high throughput pipeline. The two main parameters are the number of replicates and the type of classifier.

	SVM	**Random Forest**	**Decision Trees**	**Naive Bayes**
1	$\mathbf{0.912}s$ $0.301\pm0.016b$	$10.344s$ $0.198\pm0.054b$	$1.889s$ $0.268\pm0.021b$	$1.167s$ $\mathbf{0.352}\pm0.031b$
2	$1.597s$ $0.362\pm0.029b$	$12.346s$ $0.212\pm0.035b$	$2.747s$ $0.268\pm0.072b$	$\mathbf{1.548}s$ $\mathbf{0.399}\pm0.019b$
5	$4.384s$ $\mathbf{0.452}\pm0.010b$	$15.576s$ $0.363\pm0.028b$	$4.552s$ $0.277\pm0.074b$	$\mathbf{1.820}s$ $0.438\pm0.011b$
10	$7.073s$ $\mathbf{0.491}\pm0.014b$	$17.599s$ $0.389\pm0.016b$	$5.595s$ $0.399\pm0.009b$	$\mathbf{1.935}s$ $0.479\pm0.009b$
20	$22.335s$ $\mathbf{0.523}\pm0.008b$	$20.900s$ $0.447\pm0.015b$	$9.334s$ $0.446\pm0.026b$	$\mathbf{2.416}s$ $0.488\pm0.005b$
50	$109.636s$ $\mathbf{0.540}\pm0.008b$	$29.787s$ $0.475\pm0.012b$	$15.787s$ $0.465\pm0.007b$	$\mathbf{4.223}s$ $0.501\pm0.005b$

2	4	8
$0.055s$	$0.513s$	$0.922s$
$0.052s$	$0.520s$	$0.649s$
$0.080s$	$0.541s$	$0.680s$
$0.098s$	$0.572s$	$0.726s$
$0.156s$	$0.645s$	$1.308s$
$0.319s$	$0.845s$	$1.309s$

Table 5.1: (Left) Decoding times and MI estimates for a fixed two-state model as the **number of replicates** and the **classifier** are varied (average of five trials). (Right) Simulation times for two, four, and eight-state models as the **number of replicates** is varied (average of ten trials). Simulations were run on a machine with Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz.

With more replicates, the information lost from simulating a stochastic trajectory is minimised, resulting in better classifier performance. However, this comes with bigger a dataset, prolonging the decoding time. Similarly, some classifiers are better at certain tasks but at the expense of longer fitting times. A summary of benchmarks is listed in Table 5.1.

While SVM can classify trajectories best, it scales poorly with the number of samples and cannot fully use the additional information from more replicates in a reasonable time. Random forest scales better but is orders of magnitude far from typical simulation times, as shown in the right table. Decision trees are much faster but require far too many replicates to perform at par with the rest. Gaussian naive Bayes is the fastest and produces comparable results with SVM. We find the trade-off between near simulation and decoding times and classifier performance is best met with a naive Bayes classifier with ten trajectory replicates.

We further remark that the MI stagnates after a certain number of replicates.

5.2.3 Soundness of Estimates

By the data-processing inequality, the MI computed from a model's trajectories is bounded above by the MI computed from the nuclear concentration data of all the TFs it depends on. A simple test for the robustness of our procedure is to check that the MI calculated from, say, the concatenation of data from two TFs is not lower than the MI calculated from just one of the two. We use five TFs from the Granados et. al study [12], namely **maf1**, **mig1**, **dot6**, **tod6**, and **sfp1**. Below in Table 5.2 are the findings.

TF Group	MI	TF Group	MI
maf1	0.615	maf1, mig1, dot6	1.564
mig1	0.940	maf1, mig1, tod6	1.392
dot6	**1.107**	maf1, mig1, sfp1	1.463
tod6	0.646	maf1, dot6, tod6	1.388
sfp1	0.726	maf1, dot6, sfp1	1.423
maf1, mig1	1.197	maf1, tod6, sfp1	1.262
maf1, dot6	1.160	mig1, dot6, tod6	1.553
maf1, tod6	0.959	mig1, dot6, sfp1	**1.634**
maf1, sfp1	1.053	mig1, tod6, sfp1	1.452
mig1, dot6	**1.458**	dot6, tod6, sfp1	1.458
mig1, tod6	1.184	maf1, mig1, dot6, tod6	1.648
mig1, sfp1	1.350	maf1, mig1, dot6, sfp1	1.717
dot6, tod6	1.264	maf1, mig1, tod6, sfp1	1.621
dot6, sfp1	1.323	maf1, dot6, tod6, sfp1	1.565
tod6, sfp1	1.064	mig1, dot6, tod6, sfp1	**1.718**
		maf1, mig1, dot6, tod6, sfp1	**1.774**

Table 5.2: MI estimates for the concatenated nuclear concentration of combinations of TFs using the naive Bayes classifier. In bold are the highest MI estimates per group size.

Indeed, the MI increases with more TFs. Notably, the redundancy in the encoding between TFs can be estimated and behaves as expected, i.e. their MI, when concatenated together, is less than or equal to the sum of their individual MIs. These figures serve as a reference for the maximum MI attainable by our models, given the choice of our classifier.

Chapter 6

Experimental Setup

This chapter discusses the study's setup for optimising models by means of evolution. To begin with, we describe the data set used for the previous benchmarks and the rest of the study. We then explain how the genetic algorithm is to be structured and the genetic operators involved. Finally, we discuss the adjustments made for novelty search and propose our distance metrics for promoter models.

6.1 Exogenous Dataset

We use data from the Granados et. al [12] study to act as an exogenous input for our promoter models. In it, they perform single-cell microscopy experiments by fluorescently tagging TFs and measuring their brightness within the nucleus of yeast cells at intervals of 2.5 minutes. The cells are kept within a rich medium for at least three hours and are subjected to either carbon, osmotic, or oxidative stress for five hours. The change in fluorescence indicates the movement of TFs in and out of the nucleus.

While ten TFs were examined in the original study, only five have nuclear marker measurements that allow the proper scaling from their brightness to their concentration. These are given by **maf1**, **mig1**, **dot6**, **tod6**, and **sfp1**, all of which are considered for this study.

6.1.1 Pre-processing

The accumulation of TFs within the nucleus is evaluated by the ratio between the mean of the five brightest pixels within the cell and the median brightness of the whole cell, as was briefly tested by the same study. With this measure and the nuclear marker data, a Gaussian process is fit to predict the nuclear concentration of TFs from their fluorescence.

Our study assumes a fixed number of candidate cells, each with data on how TFs behave across different environments within them. This is not possible, however, as each cell sample is only tested for stress under one environment. To address this, we first find the minimum number of cell samples across all trials for different stress exposures, 132 cells. Then, we randomly assign to each candidate cell three unique measurements, one for each environment, from the entire dataset. We finally scale the concentrations min-max between zero and one per TF. When split across each environment, the resulting concentrations are shown in Fig. 6.1. Clear bands are present as soon as cells are subjected to stress, some occurring later or lasting longer than others.

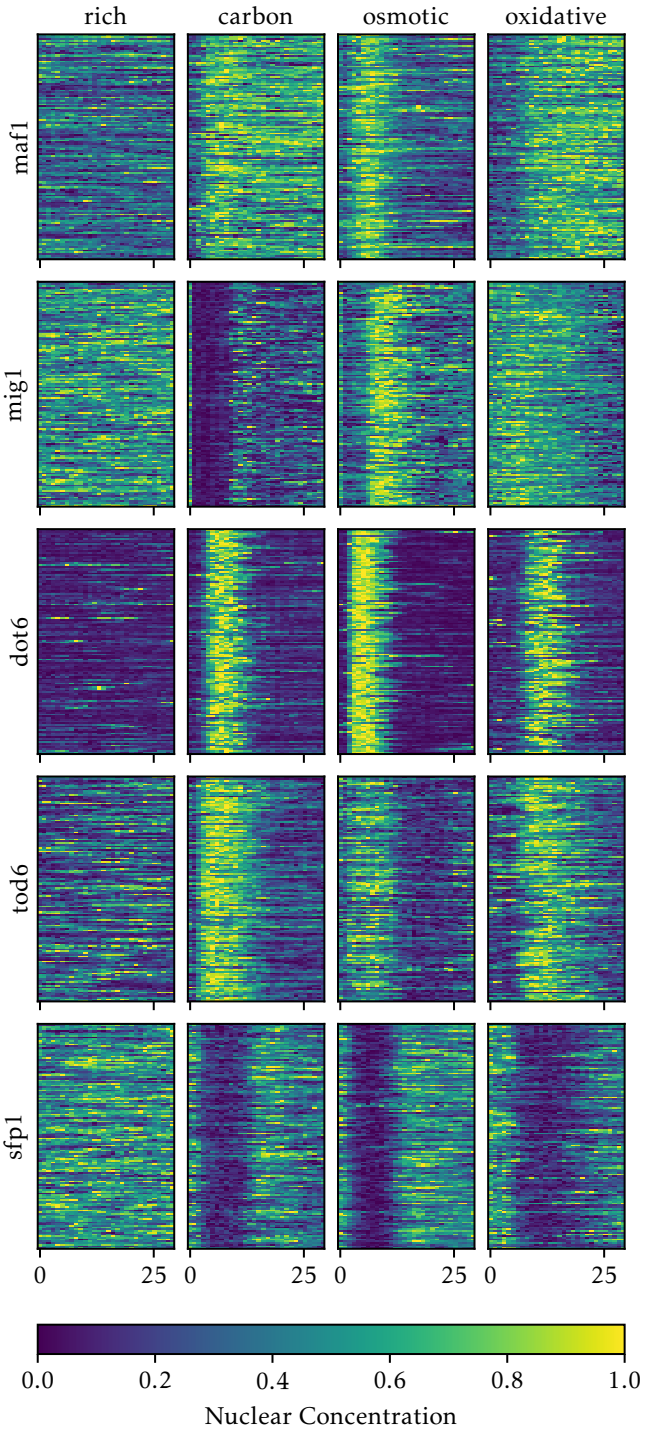

Figure 6.1: Concentrations of five TFs when introduced to different environmental conditions. The x-axis is given by time, with each interval lasting 2.5 minutes, whereas each row corresponds to a candidate cell. Note that the rich environment is down-sampled to be equal to the number of candidate cells. Contrary to the majority, notice **sfp1** leaves the nucleus when the cell is subjected to stress, whereas **mig1** exits only during carbon stress.

6.2 Genetic Algorithm

We use a traditional genetic algorithm for optimising promoter architectures. Models are assigned a fitness value equal to their MI output from the evaluation pipeline. Our operators include a non-linear crossover, a range of mutations that can happen in succession, and a conventional selection scheme. We also employ elitism to varying degrees.

6.2.1 Subgraph Swapping Crossover

We adapt the graph crossover by Globus et. al [50], originally intended for drug discovery, to satisfy the three model constraints (see 4.1.1), whereby we expect the crossover to produce two feasible offsprings given two feasible parents. While a one-point crossover on adjacency matrices was initially tested, we find that this scheme of flattening the genome does not capture the inherent characteristics of graphs, particularly with connectivity and subgraph components.

Similar to generating models, we begin by treating the model as an undirected graph, given the reversibility constraint. Each undirected edge corresponds to two directed edges with varying transition rates. Furthermore, the graph is connected and contains at least one vertex labelled as active as a consequence of the other two constraints. The two main components of this procedure are the **division of each graph into two subgraphs** and the **recombination of two subgraphs**, one from each parent, into an offspring.

Partitioning a connected graph into two connected subgraphs with the difference in their number of vertices kept to a minimum is known as the balanced connected 2-partition problem (BCP_2). It is NP-hard [78]. While studies have proposed algorithms that can achieve an approximate $4/3$ split in the number of vertices [79], we believe that the randomised split encourages more diversity within the resulting offspring.

Recall that we assume the first state of any model is active. We proceed by first choosing a random edge connecting this vertex to any of its neighbours. We remove this edge and add it to a set of *broken edges*. While we can still find a path between these two vertices, we choose a random edge in the shortest path, remove it, and add it to the broken edges. This process terminates with two connected subgraphs, possibly different in size, and a set of broken edges.

For every pair of subgraphs, one must contain the first active state. We call this the primary component and the other the secondary. The recombination stage takes a primary component from one parent and a secondary component from the other and attempts to join them together using the set of broken edges. This way, the offspring are guaranteed to have at least one active state.

Previously, we classified each edge as undirected. Now, we split it into two directed parts. We say an edge is a primary edge if it moves from a vertex in the primary component to a vertex in the secondary component. Similarly, we call its complementary edge a secondary edge. This effectively splits each set of broken edges into two, and we can assign each subgraph component its own set of broken edges, e.g. a primary component is paired with the set of broken primary edges from its original graph.

Now we consider merging two components, one primary and another secondary. We randomly select two broken directed edges, one from each of their sets, merge them back into an undirected edge, and link the two vertices these edges come from. If only broken primary

edges remain, or analogously for secondary edges, we attempt to attach each to a random vertex from any of the two components. If an edge already exists, a coin toss decides whether to replace it or discard the broken edge.

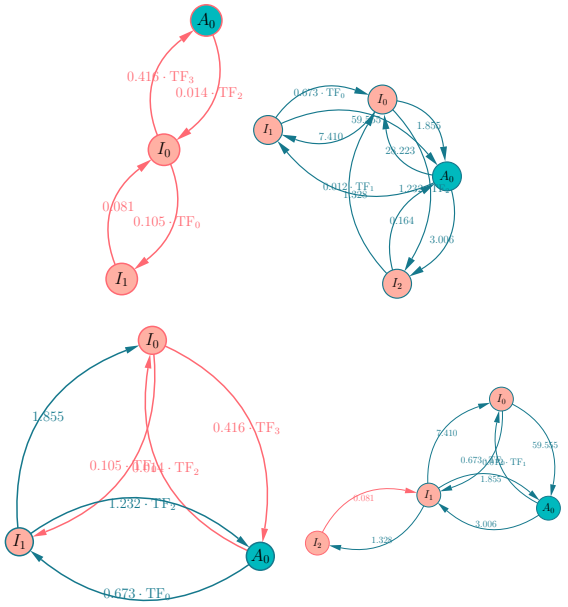

Figure 6.2: Two parent models (top row) and the two offspring they produce under the subgraph swap crossover (bottom row). Vertices and edges coming from each parent are coloured.

This results in two offspring with some genetic information from both parents. As seen in Fig. 6.2, these come in the form of possibly connected components from parents. In some domains, an optimal solution can be found by incrementally reaching stepping stones, e.g. creating efficient building blocks. This crossover allows the possibility of optimising small components and putting them together to form a larger, more effective system.

Due to the nature of the method, it does not require the two parent models to have the same number of states, as is usually the case when a uniform crossover is performed on their adjacency matrices. Hence, our method can breed any two promoter models and preserve their feasibility.

We remark that the total number of states within a population of models stays the same after a crossover. However, with mutations and selective pressure, care must be taken so models do not explode in size, as is briefed in the discussion on penalties.

6.2.2 Mutation Operators

Mutations allow models to vary their architecture enough to explore a wider search space and escape local optima. For each offspring, the following is a sequential list of mutations considered, which is applied with some probability:

1. **Edit an existing edge**: Randomly initialise a rate function to replace an existing edge. We set $p = 0.4$ for each edge.

2. **Add an edge**: Randomly add an edge between two states without direct connections. We set $p = 0.2$ for each unconnected edge.

3. **Edit a TF**: Randomly select an edge linearly dependent on a TF and change its associated TF. We set $p = 0.4$ for each linear edge.

4. **Gaussian noise on rates**: Apply Gaussian noise to the log space rate parameters. The resulting value is reflected inwards if it is beyond the bounds considered: $[-2, 2]$. We set $p = 0.8$ for each edge.

5. **Remove a vertex**: Randomly remove a vertex. To preserve connectedness, neighbouring vertices are grouped in communicating classes by union find, and each group's

randomly chosen representatives are connected. This does not apply to two-state models nor affects the first state. We set $p = 0.1$.

6. **Add a vertex**: Randomly add a vertex. To preserve connectedness, it is connected to a randomly chosen state and subsequent states are connected by some probability. Its activity is decided on a coin flip. We set $p = 0.1$ for a model to undergo this mutation and $p_{edge} = 0.25$ for any subsequent states being connected.

7. **Flip an activity**: Randomly switch active, i.e. non-zero weight, states to inactive and vice versa. In SWA, however, inactive states are given some uniformly sampled weight. This does not apply under OAS nor affects the first state. We set $p = 0.2$ for each state.

8. **Gaussian noise on activity**: Apply Gaussian noise to the activity weights of some states. We set $p = 0.8$ for each state.

We note the lack of an edge removal mutation, as this would require a bridge-locating algorithm to keep the graph connected upon removal. Bridges are edges whose removal causes a disconnection within the graph. If no bridge exists, the removal of any edge will still require the addition of another. In its absence, the vertex removal mutation is included to avoid pushing evolution towards denser graphs. By removing a vertex, at least one other edge is removed. If connectivity is unsatisfied, one additional edge is enough to resolve this, thereby causing a net loss in edge count on average.

We highlight the estimation required to set the mutation rates as a weakness of this method. Due to the number of mutations, dynamic mutation rates may be difficult to test and set up, so the rates are fixed instead.

6.2.3 Population and Selection

With multiprocessing code and 256 processors available, we consider population sizes: 500, 1000, and 1500. We set the number of iterations equal to the population size for simplicity. The initial population is randomly generated through our scheme. We choose four or eight as its initial number of states. Note, however, that the model's size across generations will vary because of the subgraph-swapping crossover. We also have three paradigms: OAS, MAS, and SWA, each run with the same set of parameters. In selecting models for the next generation, we introduce elitism with 5 or 10% of the population and use a 4-tournament selection with replacement.

We find during testing that roulette selection may be unsuitable given the arbitrary scaling of a model's MI. Also, studies have mainly employed tournament selection for multiobjective optimisation, and, to be consistent with our multiobjective goals in novelty search, we use a tournament selection for the genetic algorithm. In agreement with other studies, we use a tournament size of four, but mainly to keep selective pressure at a minimum, given the large population size.

6.2.4 Fitness Penalty Schemes

Due to the many moving parts of the setup, it may be difficult to counteract any unwanted genetic trends. For example, if the average number of states steadily increases per generation due to an imbalanced application of mutations, we want to penalise models from becoming exceedingly large. But also, if a larger number of states is what inherently produces better fitness, then we want to keep this trend in our search.

As with any modelling problem, a reduction in complexity is generally preferred. In practice, no such transcription models have been proposed to be over ten states large, for instance. To avoid this unnecessary complexity, we may employ a penalty that scales the MI proportional to its number of states or possibly even edges. During initial testing, a penalty on the number of states was employed and effectively kept the number of states below any given threshold. However, removing the penalty did not produce any larger models. We hypothesise this is attributed to going beyond the optimal architecture size and the curse of dimensionality. In large models, most parameters must be set correctly to produce decent results. For example, an inactive state could have very low transition rates, acting as an absorbing state and keeping the gene inactive throughout. A smaller model, on the other hand, may only need a few mutations before it is fully optimised.

Follow-up experiments involved penalising small models to give large models an advantage and penalising models that deviate from a specific number of states, i.e. a penalty in both directions. The findings of which are discussed in the results section.

However, we believe our mutations do not invoke bias that sways the search in any direction other than the natural trend of evolution. Hence, we do not employ any penalties for the traditional genetic algorithm.

6.3 Novelty Search with Local Competition

The genetic algorithm is adapted to operate in a multiobjective manner under the elitist NSGA-II framework [56]. The two objectives are a model's local fitness and novelty, as in novelty search with local competition (NSLC) [59], given by

$$f_{local}(\mathbf{x}) = |\{\mu_i \in \text{neighbours}(\mathbf{x}) \mid f(\mathbf{x}) > f(\mu_i)\}|, \quad \rho(\mathbf{x}) = \frac{1}{k}\sum_{i=0}^{k} d(\mathbf{x}, \mu_i)$$

where f is the fitness function, i.e. the evaluation pipeline, and each μ_i represents the model's ith nearest neighbour. Similar to Lehman's [59] study, we use $k = 15$ neighbours. All other genetic operators are carried over from the genetic algorithm setup, except we fix a population of 1000 and an elite ratio of 10%.

6.3.1 Archival Management

Unlike the conventional NSLC, we treat the archive as the end goal, akin to a quality-diversity algorithm. In our case, we maintain an unstructured archive with a fixed maximal density. Archive erosion is addressed using measures suggested in [61].

The process is as follows: after evaluating the fitness of the entire population, models are added to the archive if its novelty is above a fixed threshold θ or, given $\mu_1^\alpha, \mu_2^\alpha$—its two nearest neighbours in the archive—it ϵ-dominates μ_1^α and has a distance from μ_2^α greater than θ. We define ϵ-domination as proposed by Cully [61]: x_1 ϵ-dominates x_2 if

1. $\rho(x_1) \geq (1-\epsilon)\rho(x_2)$
2. $f(x_1) \geq (1-\epsilon)f(x_2)$
3. $(\rho(x_1) - \rho(x_2)) \cdot f(x_2) > -(f(x_1) - f(x_2)) \cdot \rho(x_2)$

Bear in mind that ϵ-domination is only used to decide whether models are archived. Pareto dominance is still employed for ranking the population.

While we initially used a traditional NSLC, the yield of good models for analysis is bounded by the elite population in the final generation, less than one hundred. After moving to a quality-diversity scheme, we tested a dynamic archival threshold and measures against archiving too many or too few models. However, some models exploded in size, as high as 36 states, due to the explorative nature of one of the novelty metrics used. This resulted in unpredictable memory consumption reaching the order of terabytes. Hence, we fix the maximal density θ and enforce a strict maximum of ten states per model—a reasonable upper bound given findings from the genetic algorithm runs. When a model is over ten states, a mutation-like operator reduces it to one of its subgraphs whose number of states is less than ten. We further note the value for θ varies across metrics.

6.3.2 Novelty Metrics

We propose two distance metrics between promoter models. Their formulations come from two separate motivations. One metric attempts to discriminate models based on their trajectory, whereas the other considers their topology.

6.3.2.1 Trajectory Metric

Looking at the very end of the pipeline, the trajectories of models are the main statistical objects used to estimate the MI. Thus, it is fair to assume that contrasting trajectories may result in different MI evaluations. While this is not true in general, such a measure can group models with similar trajectories, and local clusters can be analysed based on their trends. As such, it remains a valuable measure.

Consider any of the OAS or MAS paradigms where activity is discrete and one active state is indistinguishable from any other active state. Here, a model produces a collection of binary time series, i.e. taking values zero or one. Note that each trajectory comes from one cell only. The decoding classifier merely observes the activity of a cell and gives it a label. The classifier should not be able to access the entire cell population's continuous range of behaviours to stay true to the cell's capability of making decisions from a discrete signal. This is why no averaging has taken place for the pipeline. However, as a means for differentiation, the general behaviour of the system is well characterised precisely through a population average.

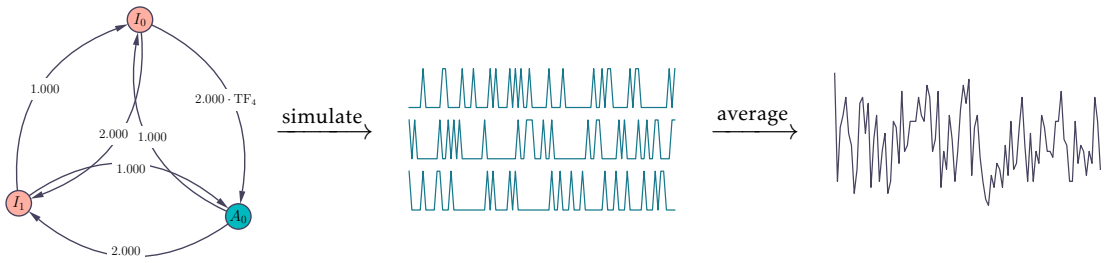

Figure 6.3: The process involved in getting the average trend of activity across all cells.

When trajectories of a model are averaged, they take a range of continuous values but can persist to be noisy. Our addition of replicates can mitigate this noise, but our choice of

replicates, 10, is already fixed to maintain an efficient pipeline. We turn to a convenient part of our matrix exponential method for a more robust average to our advantage.

Recall that we derive a probability distribution for the next state at each time step and sample a random state according to this distribution. If we keep the indicator for the state as a vector of probabilities, however, we get a trajectory of probability distributions. This is reminiscent of raising the transition matrix of a DTMC to an exponent. Given these distributions, the expected value for each time point can be calculated accordingly. As depicted in Fig. 6.4, the average trend with replicates indeed converges to this trajectory. It captures the average behaviour of the system without increasing the number of replicates or sampling any random numbers.

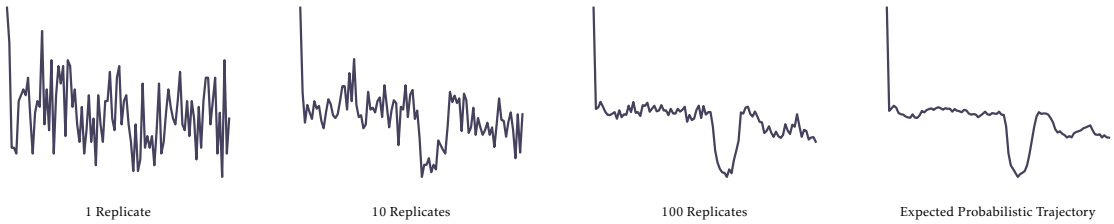

Figure 6.4: The average activity trend as the number of replicates increases. The right-most graph is the expected value of the trajectory of probability distributions derived from the matrix exponential method. The activities come from the same model in Fig. 6.3.

The Jensen-Shannon divergence is employed to compute the distance between two average trajectories, primarily for its information-theoretic interpretations and metric characteristics under a square root. A direct application does not have any desirable interpretations, however, as the activity trace is not exactly a probability distribution. It can be argued that TF dynamics involve a finite energy resource, and the distribution of energy expenditure drives its movement. However, this is not grounded in literature. We instead find the probability distribution of a state being active at each time point and compare this to the distribution at the same time point of the other trajectory. The average across all the distances is then computed, which still maintains the qualities of a metric,

$$d_{traj}(\mathbf{v}_{x_1}, \mathbf{v}_{x_2}) = \frac{1}{N_t} \sum_{i=0}^{N_t} \sqrt{d_{JS}(\mathbf{v}_{x_1}^i, \mathbf{v}_{x_2}^i)}$$

where $\mathbf{v}_x$ is the average trajectory of model x, d_{JS} is the JS-divergence, and N_t is the length of the entire time series, i.e. before any cuts. In the study, $N_t = 96$.

6.3.2.2 Topology Metric

Another possible avenue for differentiating between promoter models is through their architecture. Particularly, we are interested in the transition rates and the TFs they depend on, if any. For instance, a commonality between two models could be the existence of a shared path that passes through edges linearly dependent on **dot6**, **sfp1**, and then **tod6**. In this case, they are more likely than others to have a similar sequence of Markovian jumps. Furthermore, models that differ in the magnitude of their rates likely differ in how smooth or noisy their trajectories are. These motivate a measure describing the structural difference between promoter models.

We first convert the model into a line digraph to capture the properties and connections between edges. A line digraph is a result of converting each directed edge into a vertex and each vertex into a directed edge. Suppose we have a graph with directed edges e_{v_1,v_2} and e_{v_3,v_4} where v_i refer to vertices they connect. In its line digraph representation, there is an edge from vertex $v'(e_{v_1,v_2})$ to vertex $v'(e_{v_3,v_4})$ if and only if $v_2 = v_3$, i.e. one directed edge leads to another. Upon completion, we get a directed graph with labelled vertices.

Next, we use the Wasserstein Weisfeiler-Lehman distance metric for discrete and continuously labelled graphs [74]. Each vertex is discretely labelled by the TF it depends on—either one of the five TFs or none at all—and continuously labelled by its rate. Note that rates are distributed across the $[-2, 2]$ interval in the $\log_{10}$ space. Therefore, we take the logarithm base ten and standardise them between zero and one.

The calculation can only take place once the graphs are encoded into feature vectors. We achieve this by performing m iterations of the Weisfeiler-Lehman embedding scheme. For a graph G with vertices $v_1, \ldots, v_{|G|}$, the discrete feature vector $X_{discrete}$ is given as

$$X_{disc} = \begin{pmatrix} l^0(v_1) & l^1(v_1) & \ldots & l^m(v_1) \\ \vdots & \vdots & \ddots & \vdots \\ l^0(v_{|G|}) & l^1(v_{|G|}) & \ldots & l^m(v_{|G|}) \end{pmatrix}, \quad l^h(v) = \begin{cases} l(v) & h = 0 \\ \text{hash}(l^{h-1}(v), \mathcal{N}^{h-1}(v)) & h \geq 1 \end{cases}$$

where $l(v)$ is the original discrete label of vertex v and $\mathcal{N}^h(v)$ is the set of labels at iteration h of vertices neighbouring v. A suggested extension to continuous labels [74] is given by

$$X_{cont} = \begin{pmatrix} a^0(v_1) & a^1(v_1) & \ldots & a^m(v_1) \\ \vdots & \vdots & \ddots & \vdots \\ a^0(v_{|G|}) & a^1(v_{|G|}) & \ldots & a^m(v_{|G|}) \end{pmatrix}$$

where the continuous labels $a^h(v)$ are similarly defined as their categorical counterparts

$$a^h(v) = \begin{cases} a(v) & h = 0 \\ \frac{1}{2}\left(a^{h-1}(v) + \frac{1}{\deg(v)} \sum_{u \in \mathcal{N}(v)} a^{h-1}(u)\right) & h \geq 1 \end{cases}$$

where $a(v)$ is the original continuous label of vertex v and $\mathcal{N}(v)$ is the set of its neighbours. Now, suppose we have computed the feature vectors of line digraphs G and G' as indicated by X and X'. We compute the pairwise difference matrices D_{disc} and D_{cont} with entries

$$(D_{disc})_{ij} = \frac{1}{m} \sum_{k=0}^{m} d_{ham}\left((X_{disc})_{ik},\ (X'_{disc})_{jk}\right)$$

$$(D_{cont})_{ij} = \frac{1}{m} \sum_{k=0}^{m} d_{euc}\left((X_{cont})_{ik},\ (X'_{cont})_{jk}\right)$$

where d_{ham} and d_{euc} are the Hamming and Euclidean distances respectively. We then adapt the original method to consider both discrete and continuous labels by assigning

$$D = \frac{1}{2}\left(D_{disc} + D_{cont}\right)$$

and thus computing the 1-Wasserstein distance

$$d_{top}(X, X') = \min_{P \in \Gamma(X, X')} \langle P, D \rangle$$

where P is any matrix with row and column sums equal to $1/|G|$ and $1/|G'|$ respectively.

Note that we could have solved for the Wasserstein distances corresponding to D_{disc} and D_{cont} separately and took an average. We chose the above method for no particular reason. We also point out that we could have included the activity of states, but we decided against it for the purposes of being paradigm-agnostic and simpler in general. In agreement with the original study, we choose $m = 3$ iterations of the embedding scheme.

6.3.3 Nearest Neighbours Calculation

As with novelty search, the distance between a model and its k-nearest neighbours makes up its novelty score. Due to the metric nature of the proposed distance measures, we can employ space-partitioning data structures that take advantage of metric space properties.

For efficiency reasons, we pre-compute the feature vectors of promoter models and use them for distance calculations. For the trajectory metric, feature vectors come in arrays of length 96. Metric spaces with elements that can be expressed as vectors benefit from algorithms where, for example, midpoints can be calculated between two arbitrary points. We use ball trees to compute our trajectory distances.

For the topology metric, feature vectors come in arbitrary-sized 2D matrices with width m. While these could be flattened to attain a vector representation, they lose their structure and therefore lead to incorrect pruning of the search space. As such, we treat it as an arbitrary metric space where only the distances between two points and not their representations are accessible. Vantage point trees, M-trees, and cover trees are data structures that allow for this. However, due to the high intrinsic dimensionality of the space, it is not as good as a brute-force approach taking symmetry into account. For this reason, we resort to multiprocessing for speed gains.

A simple test to confirm our reasoning is by estimating the intrinsic fractal dimensionality of the metric space. We use the method of manifold-adaptive local dimension estimation [65]. To begin with, we randomly generate 2000 models with states that are uniformly likely to be anywhere between two to ten. Then, their pairwise distance matrix is computed using the topology metric and estimated its intrinsic dimensionality. We find estimates of about 80. This makes sense as we have two labels per edge, and a fully connected ten-state model has 45 edges, amounting to at most 90 dimensions.

Chapter 7

Results and Discussion

In this chapter, we highlight and discuss findings from the evolutionary search. We begin by examining results from the genetic algorithm and the role of certain parameters of the setup that contribute to their trends. Then, we present the best models derived from our method and analyse them through novelty search results. Follow-up experiments to support our hypotheses are also mentioned.

7.1 Evolutionary Trends and Findings

This section primarily focuses on the genetic algorithm. A total of 36 runs were performed, each representing different parameter combinations. Due to this large number, we only present snapshots of the results where necessary.

7.1.1 Maximum Fitness

The genetic algorithm found models with MI as high as 1.3 bits as listed in Table 7.1. As expected, larger population sizes allow more models to be considered and thus enable it to find better models on average. Regardless of size, however, we note that some runs appear to converge to local optima. For example, MAS runs with 1500 models do not necessarily find the best models despite their high average fitnesses. This could be the case of the elite population converging to a similar non-optimal but fit architecture. We explore this hypothesis later. Another possibility is the selective pressure being too high, as backed by OAS runs with 1500 models finding better populations with a lower elite ratio.

The most important extrapolation from these results is the comparable fitnesses found for all three paradigms, despite their increasing generality. We find that the simplest paradigm, OAS, is capable of encoding signals just as well as the other two. This is supported by current literature where models proposed typically possess only one active state and where the consensus is that gene activity is considered discrete. However, another possibility is the high dimensionality of the more general paradigms, limiting the likelihood of finding the best models. We later examine the architecture of good models from MAS and SWA and see if they bear any resemblance with the simplest paradigm.

As with each run, statistics were recorded for every generation. The elites' fitness trends are plotted in Fig. 7.1. The average fitness appears to stagnate past a few hundred iterations, and OAS models are optimised quicker than the rest.

Population \ Paradigm		One Active State		Multiple Active States		Spectrum of Weighted Activities	
500	5% Elite	1.221*b* 1.181*b*	1.231*b* 1.166*b*	1.195*b* 1.108*b*	1.165*b* 1.088*b*	1.244*b* 1.185*b*	1.217*b* 1.153*b*
	10% Elite	1.295*b* 1.181*b*	1.275*b* 1.189*b*	1.265*b* 1.190*b*	1.205*b* 1.124*b*	1.284*b* 1.141*b*	1.237*b* 1.113*b*
1000	5% Elite	1.289*b* 1.222*b*	1.267*b* 1.218*b*	**1.318b** 1.177*b*	1.268*b* 1.171*b*	1.277*b* 1.203*b*	1.236*b* 1.182*b*
	10% Elite	1.316*b* 1.248*b*	**1.327b** 1.226*b*	1.314*b* 1.187*b*	1.295*b* 1.194*b*	1.264*b* 1.200*b*	1.272*b* 1.177*b*
1500	5% Elite	**1.340b** 1.247*b*	**1.327b** **1.259b**	1.233*b* 1.174*b*	**1.328b** 1.196*b*	1.286*b* 1.206*b*	1.289*b* 1.200*b*
	10% Elite	1.339*b* **1.255b**	1.295*b* 1.213*b*	1.298*b* **1.224b**	1.300*b* **1.216b**	**1.322b** **1.217b**	**1.294b** **1.214b**
Initial No. of States		4	8	4	8	4	8

Table 7.1: Maximum and average fitness of the elite population found by the evolutionary search under certain parameters. Note that the number of iterations is equal to the population size. The largest fitness values per column are in bold.

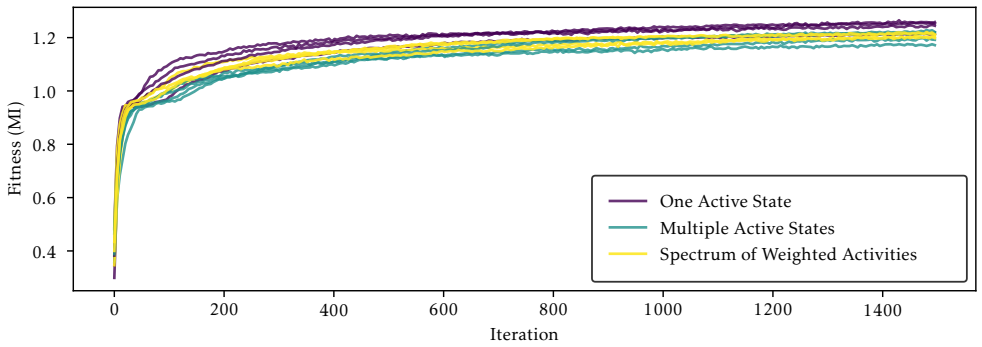

Figure 7.1: The trend for the average fitness of the elite population for all 12 runs with a population of 1500. Lines coloured by activity paradigm.

7.1.2 Convergence in Size

Fig. 7.2 similarly shows the trends of the average number of states for the elite population of the same 12 runs. Regardless of the initial population, the average number of states converges to around three to five. We believe this is an indication of the genetic algorithm converging to an optimal set of architecture sizes. We believe models with any lesser states are too simple to capture complex behaviour, and those with significantly more states produce trajectories that are too noisy to be reliably decoded.

Simpler paradigms have slightly more states on average. This is reasoned by the capability of activity weights or the presence of another active state to encode more information with fewer states. The next section illustrates this fact.

A noticeable trend also occurs at the start of every run. For populations with an initial number of states equal to eight, the average number of states drops significantly within the first hundred generations. The case is similar but less pronounced for populations that begin

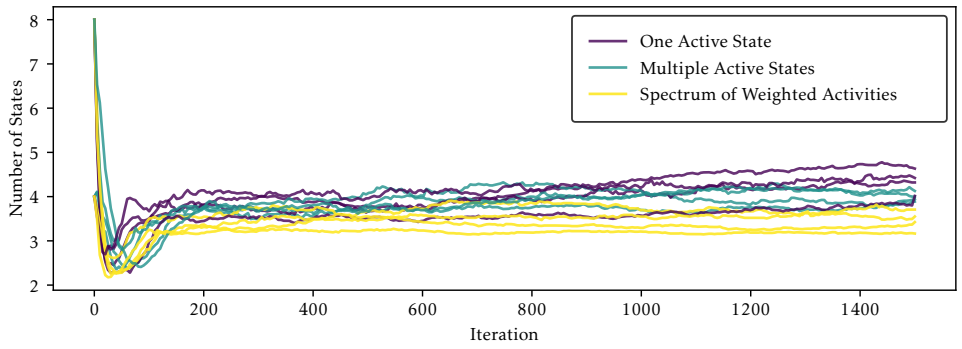

Figure 7.2: The trend for the average number of states of the elite population for all runs with a population of 1500. Lines coloured by activity paradigm.

in the four-states mark. We believe this is attributed to the curse of dimensionality involved with optimising weights and connections for a large Markov chain. Furthermore, we can draw parallels to stepping stones, as similarly pointed out by Lehman [59]. The crossover operator assigns each offspring a subgraph component from each parent. As small models become quite optimised, the offspring in later generations inherit efficient components from their parents, possibly making them fitter with size. Just as soon as the 100th iteration passes, the average number of states begins to increase, but only past a certain point.

7.1.3 Reversed and Balanced Penalties

While the above genetic algorithm runs have no penalties, we initially explored their integration to control the genetic trends. As depicted in Fig 7.2, the search naturally steers itself towards a number of states rather than monotonically increasing with iteration. This suggests there is no need for penalising large models.

However, we previously mentioned that the curse of dimensionality might be preventing large models from being optimised fast enough, leading to the elite population being filled with smaller models and successive generations following suit. To verify this claim, we consider penalising smaller models to steer the search towards larger models. We further consider a penalty for models deviating from a number of states. We call these the reversed and balanced penalties, respectively and define them as follows

$$\text{penalty}_{rev}(x) = 2 - 2 \cdot \exp\left\{-\left(\frac{1}{m}\max\{0, N - states(x)\}\right)^n\right\}$$
$$\text{penalty}_{bal}(x) = 2 - 2 \cdot \exp\left\{-\left(\frac{1}{m}\left(states(x) - states_{target}\right)^2\right)^n\right\}$$

Two is chosen as it is the maximum MI attainable given four environmental conditions to discriminate against. Here, m controls the horizontal scaling and n controls the steepness of the penalty. In our case, we set $m = 8$ and $n = 6$. Note that it is applied as

$$f(x) = \max\{0, \text{MI}(x) - \text{penalty}(x)\}$$

Shown in Fig. 7.3 are such trends for these two penalty schemes. It appears that the balanced penalty successfully keeps the average number of states within a small interval about the target number of states. Similarly, the reversed penalty converges to some number of states.

However, as with both schemes, their fitnesses are subpar from those found from the original search with no penalties. Specifically, they either stagnate early or take too long to get optimised. While further testing is required, we do believe models that are any larger

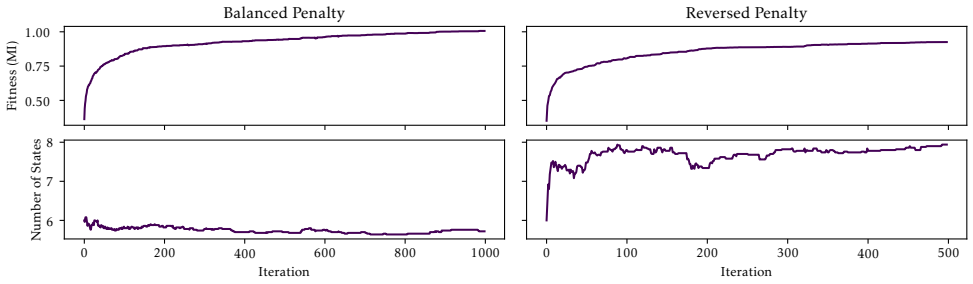

Figure 7.3: The trend for the average fitness and number of states for genetic algorithms with a balanced penalty centred at six states and a reversed penalty, both with $m = 8$ and $n = 6$

produce trajectories that are more prone to noise due to their number of states.

7.1.4 Visualising the Searched Space

An initial hypothesis was that certain runs were converging to local optima, with elites becoming nearly identical in architecture. We may find the pairwise distances between models in the entire population using our topology metric to verify this. Fig. 7.4 presents two visualisations for the resulting population of a genetic algorithm run. Indeed, the elites

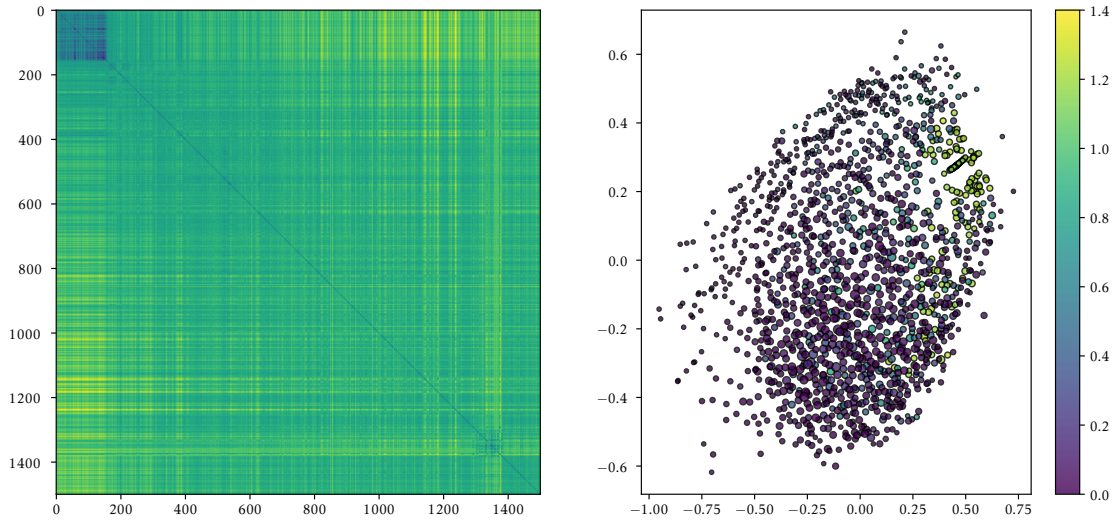

Figure 7.4: (Left) the distance matrix under the topology metric and (Right) its corresponding multi-dimensional scaling (MDS) plot with two components. Each point in the MDS plot corresponds to a model coloured by their fitness. The population is derived from the genetic algorithm run under OAS with a population of 1500, 10% of which are elites, and eight as the initial number of states.

are similar in architecture. This is indicated by the dark region in the upper-left corner of the distance matrix and the points closely clustered together in the MDS plot. This is a primary motivation for our novelty search setup, whereby a collection of fit and diverse

solutions are maintained. The MDS plot further demonstrates the unevenness of the search space covered, with dense and sparse regions throughout. Our novelty search maintains a fixed density within the archive.

7.2 Characteristics of High MI Models

This section analyses models from the evolutionary search with exceptionally high MI.

7.2.1 The Promoter Hall of Fame

We highlight the best models found under each activity paradigm. As certain architectures were extremely common, we selectively chose to present diverse examples and, whenever necessary, mention their high frequency, a possible indication of their faithfulness to reality. In the following diagrams, TFs are indexed from zero and are given in order by: **maf1**, **mig1**, **dot6**, **tod6**, and **sfp1**.

We first consider models with one active state. Across all the runs, most elites have a chain-like architecture with three, four, or five states, with one end being the active state. This structure is exhibited by the first two models in Fig 7.5. A commonality between all chain-like architectures is the pairs of TFs that each reversible pair of directed edges correspond to. We typically find pairs such as: **maf1** and **sfp1**; **mig1** and **tod6**; and **mig1** and **dot6**, with the first two occurring between two inactive states and the latter between an active state and an inactive state. All the models in Fig. 7.5 have this reversible edge of **mig1**

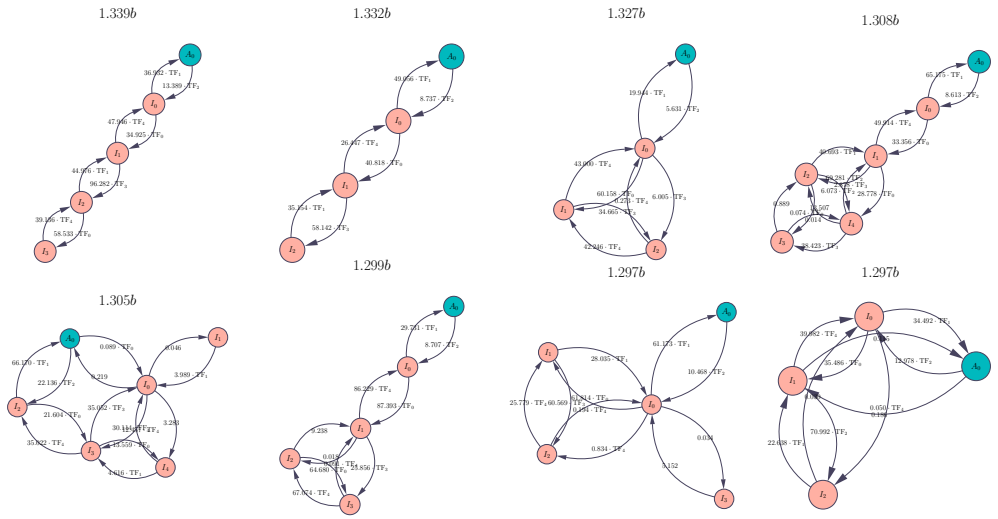

Figure 7.5: Models under the OAS paradigm with high MI. The first and second models listed represent more than 50% of the best models, only differing in the permutations of specific TF pairs making up each edge.

and **dot6** connecting to the active state, arguably the most important edge for OAS models. We briefly discuss this phenomenon in the next subsection.

Moving onto models with multiple active states, similar pairs of TFs can be observed, as shown in Fig. 7.6. Chain-like architectures also dominate the elite population but with considerably fewer five-state chains. Visibly, there is a resemblance to OAS models in that

active states tend to be grouped. Transitions between active states follow the same pairs of TFs as previously between two inactive states.

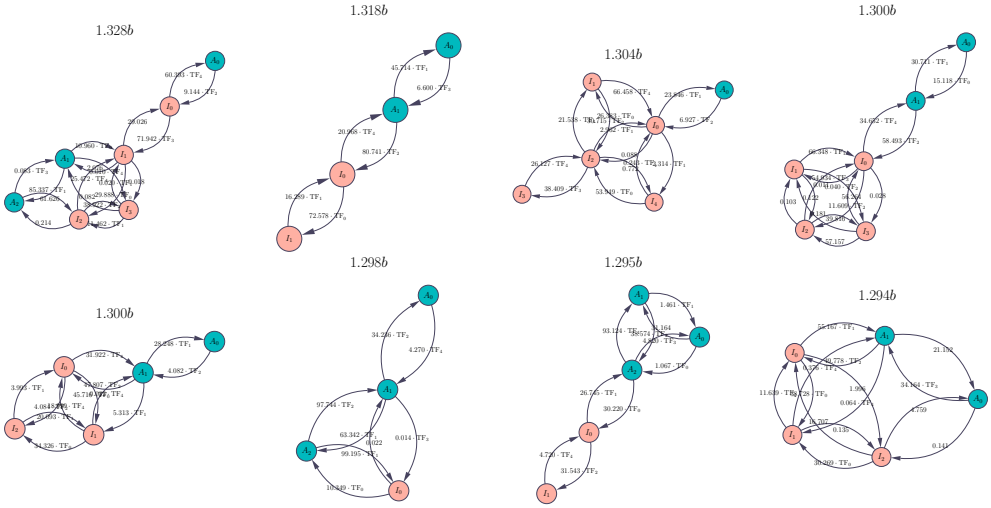

Figure 7.6: Models under the MAS paradigm with high MI. Active states emerge in groups.

Finally, models with weighted activities are depicted in Fig. 7.7. Unsurprisingly, they are much simpler in architecture than the previous two. Once again, chain-like structures make up most of the top models—this time, however, with the appearance of three-state chains. The distribution of weights across the states favours the grouping of active states in clusters like in MAS. Some models have nearly identical weights, and some have weights forming a gradient to produce smoother, more interpolated trajectories.

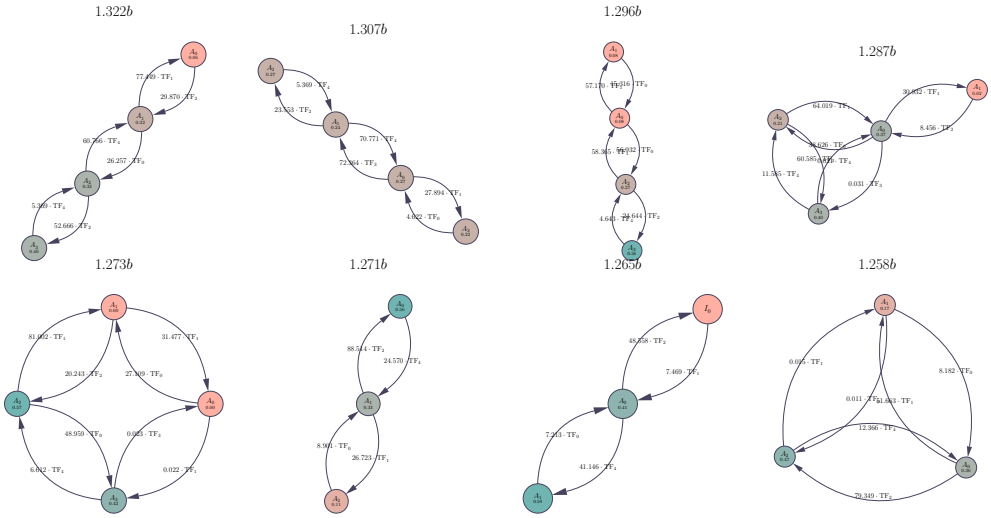

Figure 7.7: Models under the SWA paradigm with high MI. Architecture is smaller, but the spectrum of activities more than makes up for it in encoding information.

7.2.2 Blueprints for an Effective Architecture

We ran a novelty search with a population of 1000 under the topology metric. The result was an archive of 1489 fit and diverse, uniformly interspersed models. From this, we aim to find common structural characteristics that make up a good model. Here, we focus on the topology metric for its direct involvement with a model's architecture.

Contrary to the genetic algorithm results, the archive is spaced more uniformly, as indicated by the MDS plot in Fig. 7.8. However, we note that models in the archive have good average MI but not anywhere on the level of the maximum MI found by the genetic algorithm. The average archive fitness came around to about 0.691, whereas the entire population (including non-elites) for the genetic algorithm we have considered is 0.280. We believe this is the case of architecture weights not being optimised fully, as the search is balancing multiple objectives at once. With a larger number of generations, the archive should further improve performance, whereas that of the traditional genetic algorithm converges prematurely. We maintain, however, that the connections and relationships between TFs within the archive remain important.

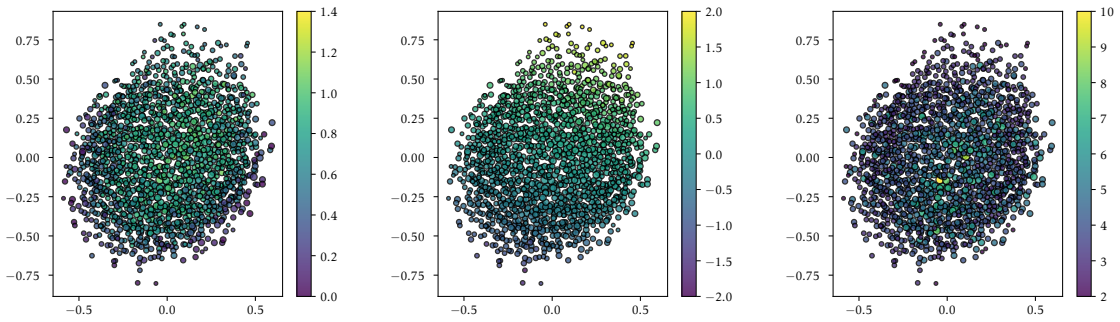

Figure 7.8: MDS plot of models from the novelty archive with distances calculated by the topology metric and points highlighted based on (Left) fitness, (Middle) average logarithmic rates, and (Right) number of states. The archive is from an NSLC run under the OAS paradigm.

Note that the topology metric captures the subtleties within the model's architecture, as indicated by the clear patterns when points are highlighted based on their average rates or number of states. From these plots alone, we can cross-reference the areas with the best fitness with those of the other two plots to see what characterises such models, e.g. average logarithmic rates are near zero. Even so, we are more interested in the dynamics of TFs and their relationships that form a good model.

We have seen in the previous subsection TFs that come in pairs. If we were to analyse their appearance in the results of genetic algorithms, however, there is some bias induced by the selective pressure and cross-breeding between elites that skews the figures to some fixed architecture. As such, we bring our analysis to the archive found.

Measuring the frequency of pairs of edges, we arrive at the histogram in Fig. 7.9. While the archive is constantly replaced with better models during the search, if the number of iterations is too short, the "borders" of the archive may still be underdeveloped. We separate the models into three classes, increasing in fitness, as an additional measure. A clear preference for **dot6** and **sfp1** is suggested by our findings. This pair does not have the highest mutual information together, a title belonging to **mig1** and **dot6**—the second most frequent pair to appear. However, looking through their nuclear traces as in Fig. 6.1, we find

they have a nearly exact complementary set of nuclear traces. We say complementary in the sense that when one TF enters the nucleus, the other leaves. The case is similar for **tod6** and **sfp1**, but **dot6** has a less noisy trace. Despite being inconclusive, the slight preference for **dot6**, when paired with a constant rate, may also be attributed to its clearer signals.

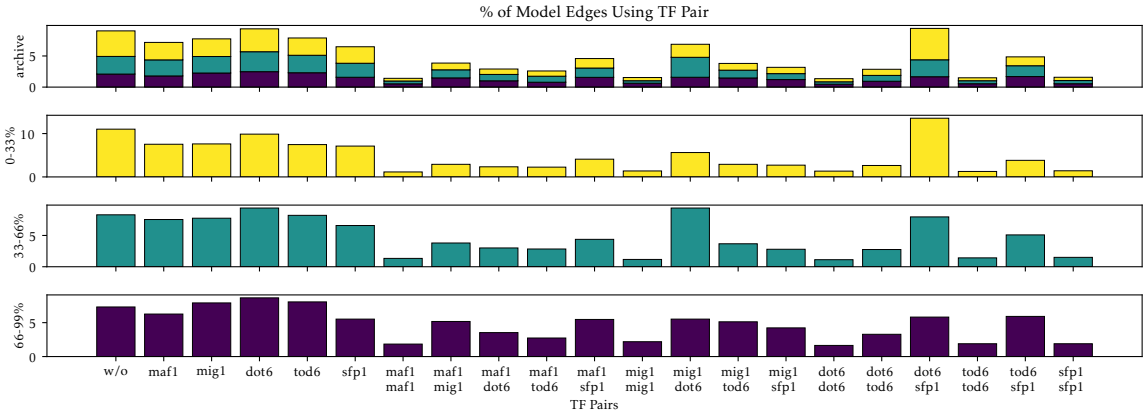

Figure 7.9: The percentage of model edges using some pair of TFs to connect between two states. The top row is representative of the entire archive, whereas the bottom three are different equally sized echelons of fitness. For example, the second row comprises the top 33% models. Note the scale of the second row is twice as large as the bottom two.

On the other hand, we also analyse the TF groups the models take as input in relation to the maximum MI attainable. A similar fitness-based division is performed, shown in Fig. 7.10. As expected, incorporating all the TFs admits the highest MI ceiling. However, choosing TFs with a large collective MI does not guarantee a sound architecture. Here, again, **dot6**

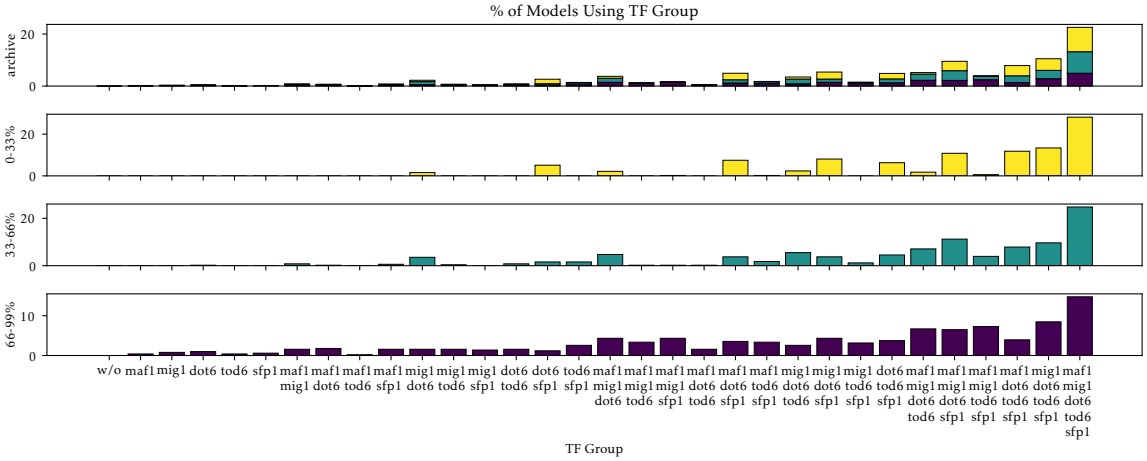

Figure 7.10: The percentage of models that depend on a group of TFs, i.e. have rates dependent on them. Similarly, the top row is representative of the entire archive, and the bottom three are the top models within their corresponding brackets.

and **sfp1** are favoured over pairs with more information. Another example is the pairing of **maf1**, **dot6**, and **tod6** appear to be underutilised. Although theoretically high in MI, neither TF complement the other. As a rule of thumb, the best TFs to use are those with suitable complementary pairings and have few redundancies in their environmental encoding.

7.2.2.1 The Chain Architecture

We found that the genetic algorithm often converged to models with a chain-like architecture, utilising a particular set of TF pairs to move between two given states. This linear structure often has an active state on one of its ends. We attempt to understand it from the context of models in the simplest paradigm—those with one active state.

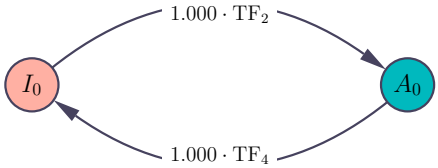

Figure 7.11: A two-state model with an edge pair depending on **dot6**, indexed by two, and **sfp1**, indexed by four. These TFs have complementary nuclear traces during stress.

In the previous subsection, we found the best TF pairings to be selected for their complementary behaviour. Consider a small two-state model utilising two complementary TFs in **dot6** and **sfp1** as in Fig. 7.11. For their traces, we refer the reader to Fig. 6.1. In the event of carbon stress, **dot6** accumulates within the nucleus, making it extremely likely for the system to be in the active state. While in the active state, we only move back to the inactive state once the concentration of **sfp1** returns to normal. If, by chance, we move back to the inactive state before this, the likelihood of returning to the active state remains high. Thus the system stays in the active state until the concentrations stabilise. This chain acts as a noise suppression mechanism, keeping the activity from rapidly switching.

Now, consider when subunits are chained together. We bring up the best model found in the one-active state paradigm as in Fig. 7.12. Here, the pairs of TFs in between states are complementary in one environment. While not apparent at first, notice that at any inactive state, the shortest path to the active state relies on the concentration of TFs: **mig1** and **sfp1**, whereas the active state moves to any of the inactive states via paths relying on the TFs: **maf1**, **dot6**, and **tod6**. Analysing their nuclear concentration data shows that these two groups are strongly complementary in the carbon stress environment and somewhat complementary in the other two environments. In fact, analysing the linear paths of other found chain models confirms the same set of these two groups.

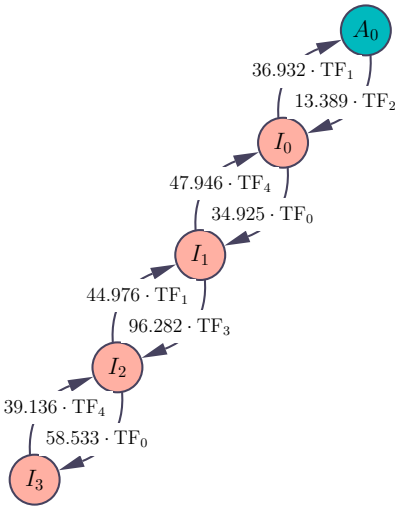

Figure 7.12: The best chain found by the genetic algorithm with one active state.

In the event of carbon stress, for example, this model moves as far away as possible from the active state, reaching the inactive state at the end. While the stochastic system could still make jumps to the next state, chaining these inactive states makes it unlikely to reach the active state until the concentrations go back to normal. With this in mind, we can attribute the length of the inactive state chain almost as a lagging mechanism before the gene is switched back to active. Short chains may not be enough to prevent the gene from moving to the active state, and extremely long chains may keep the gene inactive for too long. As such, we find chain models with bounded lengths.

A natural extension is to likewise chain the active states. This behaviour is exhibited by some models found by the genetic algorithm under the MAS paradigm.

7.2.3 Dynamics of Activity Trajectories

Recall that the dynamics of a model are partially captured by its average trajectory. Consider finding the pairwise trajectory distances between models in the same genetic algorithm run as previously. Fig. 7.13 depicts the resulting distance matrix and MDS plot.

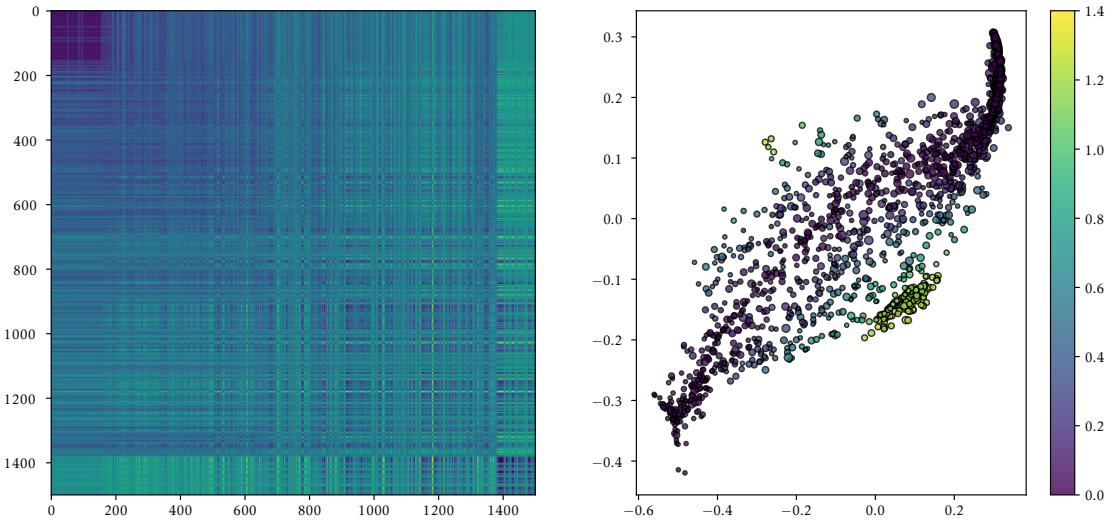

Figure 7.13: (Left) the distance matrix under the trajectory metric and (Right) its corresponding MDS plot with two components. Each point in the MDS plot corresponds to a model coloured by their fitness. The population is derived from the genetic algorithm run under OAS with a population of 1500, 10% of which are elites, and eight as the initial number of states.

We find the trajectory metric can quantify the similar behaviour of elites, as is apparent with the dark upper-left region of the matrix. However, unlike before, we find two diametrically opposite clusters in the MDS plot. Taking one point from each cluster and calculating their trajectories, we see they are essentially flipped in terms of activity as in Fig. 7.14.

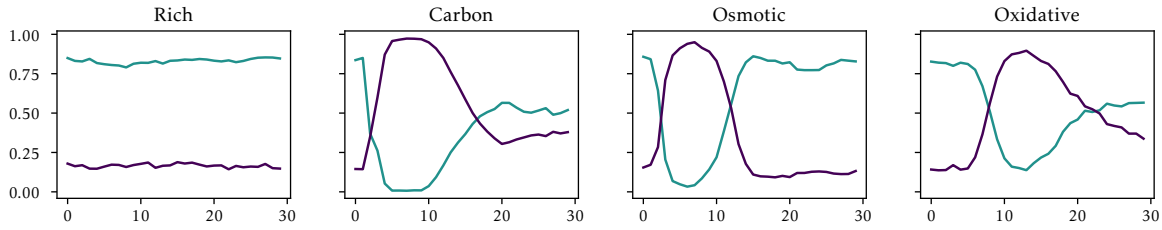

Figure 7.14: Average trajectories of two models belonging to one of the two opposite clusters in the MDS plot.

From the point of view of the machine-learning decoder, flipping the activity signal makes no difference to the amount of information it possesses. With this symmetry, one can immediately interpret the behaviour of other regions within the MDS plot. The two ends of the spiral are the trivial trajectories, with the gene either always active or inactive, and those

in the middle are somewhere between the flipped versions of the elite trends, e.g. possessing smaller peaks. Sampling models from these regions confirm this. Most high MI models share the same trends shown in Fig 7.14, which resemble the nuclear concentration data of TFs: **dot6**, **tod6**, and **sfp1**. This makes sense given that chain-like architectures are heavily dependent on the concentrations of these TFs to move between states.

We initially thought that the models' quality of having average trajectories similar to TFs nuclear traces was a local optimum. In search of more non-trivial trajectories, we seek to evolve models purely based on their trajectories. Results from the novelty search under the trajectory metric yielded an archive of 1010 models. An MDS plot is given in Fig. 7.15. Models sampled from the extreme ends produce the exact same trajectories as above. We hypothesise that this trend is likely optimal and is a result of the TFs accessible. With the addition of more TFs diverse in behaviour, e.g. oscillatory, we may find a different optimal trajectory.

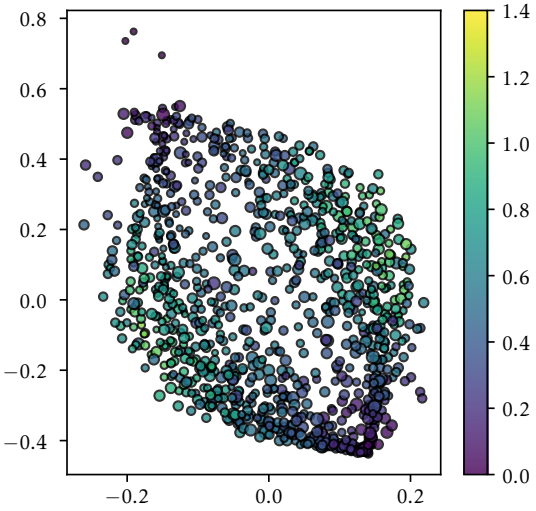

Figure 7.15: MDS Plot of models from the novelty archive with distances calculated by the topology metric. The archive is from an NSLC run under the OAS paradigm.

Chapter 8

Conclusion

In this study, we formalised promoter models and developed a methodology to optimise them for encoding extracellular information.

We framed promoter models as time-inhomogeneous continuous-time Markov chains and considered three increasingly general paradigms for gene activity. We showed that one active state models can produce comparable mutual information estimates to the other two, with chain-like architectures being the most successful. Multiple active state models were often found to have active states connected in components, resembling a large active state. Models with a spectrum of weighted activities resulted in simpler architectures but appeared more challenging to optimise than the other two due to their significantly higher dimensionality.

We presented a matrix exponential approximation for simulating promoter models and showed it is orders of magnitude faster yet produces similar trajectories with the Gillespie algorithm for time-dependent propensities. We also showed how it could be adapted to simulate the average activity trend for a model quickly. We remark, however, that our method becomes resource expensive when the number of states is considerably high.

To complete the evaluation pipeline, we pursued a trade-off between the number of replicates and the type of machine learning classifier to keep simulation and decoding times within the same order of magnitude with as high accuracy as possible. Despite this, further improvements could be made to the machine-learning pipeline, particularly in choosing a framework that can better capture the temporal attributes of the time series data. The TFs we currently use exhibit simple dynamics during stress. For example, when oscillatory TFs are introduced, a more sophisticated decoder may be required to maintain a high lower bound estimate for MI.

We employed genetic algorithms with a set of operators that preserve the feasibility of models and demonstrated they could successfully find models with high MI. However, we mention its orchestration required incredible effort and testing.

Finally, we proposed two distance metrics between promoter models, one looking at its topology and another at its trajectory. We showed both are useful tools for visualising the model search space and performed a novelty search with local competition to evolve an archive of fit and diverse models for which we analysed extensively.

8.1 Future Work

We chose to decouple mRNA from the model for a variety of reasons. However, once a model has been optimised, we could reintroduce mRNA, albeit at the cost of having to use an SSA. The trends of mRNA could then be compared with what might be expected in practice, as is usually the process for transcriptional modelling studies.

We mentioned previously the oscillatory nature of some TFs. Our method has yet to be extensively tested on it. An interesting setup is to allow the genetic algorithm access to multiple TFs, some incredibly noisy, some with clear peaks, and some oscillatory. The method can thus be evaluated by its ability to use certain TFs to its advantage. For example, we found earlier that complementary TFs are often linked. We could test if a phase shift in oscillations is able to mimic this noise reduction technique.

While a few tests have been run, we have not extensively studied the resulting architecture sizes when the set of TFs is constrained. Further work could be done to establish whether the model sizes depend on the number of TFs or if an upper limit exists to which the model becomes too noisy.

We believe other studies can benefit from the methods within this paper and hope to make the current repository more accessible.

Bibliography

[1] Bruce Alberts, Dennis Bray, Karen Hopkin, Alexander D Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Essential cell biology*. Garland Science, 2015.

[2] Harley H McAdams and Adam Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences*, 94(3):814–819, 1997.

[3] Greg Gibson. The environmental contribution to gene expression profiles. *Nature reviews genetics*, 9(8):575–581, 2008.

[4] Rune Linding and Edda Klipp. Shapes of cell signaling. *Current Opinion in Systems Biology*, 27:100354, 2021.

[5] Juan M Pedraza and Alexander van Oudenaarden. Noise propagation in gene networks. *Science*, 307(5717):1965–1969, 2005.

[6] Nan Hao and Erin K O'shea. Signal-dependent dynamics of transcription factor translocation controls gene expression. *Nature structural & molecular biology*, 19(1):31–39, 2012.

[7] Susan Y Chen, Lindsey C Osimiri, Michael Chevalier, Lukasz J Bugaj, Taylor H Nguyen, RA Greenstein, Andrew H Ng, Jacob Stewart-Ornstein, Lauren T Neves, and Hana El-Samad. Optogenetic control reveals differential promoter interpretation of transcription factor nuclear translocation dynamics. *Cell systems*, 11(4):336–353, 2020.

[8] Martijn Rep, Jacobus Albertyn, Johan M Thevelein, Bernard A Prior, and Stefan Hohmann. Different signalling pathways contribute to the control of gpd1 gene expression by osmotic stress in saccharomyces cerevisiae. *Microbiology*, 145(3):715–727, 1999.

[9] Jimmy Vandel, Océane Cassan, Sophie Lèbre, Charles-Henri Lecellier, and Laurent Bréhélin. Probing transcription factor combinatorics in different promoter classes and in enhancers. *BMC genomics*, 20(1):1–19, 2019.

[10] Gregor Neuert, Brian Munsky, Rui Zhen Tan, Leonid Teytelman, Mustafa Khammash, and Alexander Van Oudenaarden. Systematic identification of signal-activated stochastic gene regulation. *Science*, 339(6119):584–587, 2013.

[11] Edward Tunnacliffe and Jonathan R Chubb. What is a transcriptional burst? *Trends in Genetics*, 36(4):288–297, 2020.

[12] Alejandro A Granados, Julian MJ Pietsch, Sarah A Cepeda-Humerez, Iseabail L Farquhar, Gašper Tkačik, and Peter S Swain. Distributed and dynamic intracellular organization of extracellular information. *Proceedings of the National Academy of Sciences*, 115(23):6088–6093, 2018.

[13] Kamil Bobrowski. Modelling promoter activity in *Saccharomyces Cerevisiae*. Master's thesis, Imperial College London, 2022.

[14] Francis Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

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

[16] Michel Jacquet, Georges Renault, Sylvie Lallet, Jan De Mey, and Albert Goldbeter. Oscillatory nucleocytoplasmic shuttling of the general stress response transcriptional activators msn2 and msn4 in saccharomyces cerevisiae. *The Journal of cell biology*, 161(3):497–505, 2003.

[17] Harley H McAdams and Adam Arkin. It's a noisy business! genetic regulation at the nanomolar scale. *Trends in genetics*, 15(2):65–69, 1999.

[18] Eric Davidson and Michael Levin. Gene regulatory networks. *Proceedings of the National Academy of Sciences*, 102(14):4935–4935, 2005.

[19] Nabil Guelzim, Samuele Bottani, Paul Bourgine, and François Képès. Topological and causal structure of the yeast transcriptional regulatory network. *Nature genetics*, 31(1):60–63, 2002.

[20] Arjun Raj and Alexander Van Oudenaarden. Nature, nurture, or chance: stochastic gene expression and its consequences. *Cell*, 135(2):216–226, 2008.

[21] Richard Durrett and R Durrett. *Essentials of stochastic processes*, volume 1. Springer, 1999.

[22] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[23] 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(3):e240311, 2014.

[24] Andrey Kolmogorov. On the shannon theory of information transmission in the case of continuous signals. *IRE Transactions on Information Theory*, 2(4):102–108, 1956.

[25] Thomas M Cover, Joy A Thomas, et al. Entropy, relative entropy and mutual information. *Elements of information theory*, 2(1):12–13, 1991.

[26] Janett Walters-Williams and Yan Li. Estimation of mutual information: A survey. In *Rough Sets and Knowledge Technology: 4th International Conference, RSKT 2009, Gold Coast, Australia, July 14-16, 2009. Proceedings 4*, pages 389–396. Springer, 2009.

[27] Jonathan R Chubb and Tanniemola B Liverpool. Bursts and pulses: insights from single cell studies into transcriptional mechanisms. *Current opinion in genetics & development*, 20(5):478–484, 2010.

[28] Georg Rieckh and Gašper Tkačik. Noise and information transmission in promoters with multiple internal states. *Biophysical journal*, 106(5):1194–1204, 2014.

[29] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[30] David F Anderson. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *The Journal of chemical physics*, 127(21):214107, 2007.

[31] Margaritis Voliotis, Philipp Thomas, Ramon Grima, and Clive G Bowsher. Stochastic simulation of biomolecular networks in dynamic environments. *PLoS computational biology*, 12(6):e1004923, 2016.

[32] Muruhan Rathinam, Linda R Petzold, Yang Cao, and Daniel T Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24):12784–12794, 2003.

[33] Yang Cao, Daniel T Gillespie, and Linda R Petzold. Avoiding negative populations in explicit poisson tau-leaping. *The Journal of chemical physics*, 123(5):054104, 2005.

[34] Tatiana T Marquez-Lago and Kevin Burrage. Binomial tau-leap spatial stochastic simulation algorithm for applications in chemical kinetics. *The Journal of Chemical Physics*, 127(10):09B603, 2007.

[35] Jill Padgett and Silvana Ilie. An adaptive tau-leaping method for stochastic simulations of reaction-diffusion systems. *AIP Advances*, 6(3), 2016.

[36] Asger Hobolth and Eric A Stone. Simulation from endpoint-conditioned, continuous-time markov chains on a finite state space, with applications to molecular evolution. *The annals of applied statistics*, 3(3):1204, 2009.

[37] Ying Tang, Adewunmi Adelaja, Felix X-F Ye, Eric Deeds, Roy Wollman, and Alexander Hoffmann. Quantifying information accumulation encoded in the dynamics of biochemical signaling. *Nature communications*, 12(1):1–10, 2021.

[38] Sarah Anhala Cepeda-Humerez, Jakob Ruess, and Gašper Tkačik. Estimating information in time-varying signals. *PLoS computational biology*, 15(9):e1007290, 2019.

[39] Ying Tang and Alexander Hoffmann. Quantifying information of intracellular signaling: progress with machine learning. *Rep. Prog. Phys*, 85(086602):086602, 2022.

[40] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.

[41] H Holland John. Adaptation in natural and artificial systems. *Ann Arbor: University of Michigan Press*, 1975.

[42] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80:8091–8126, 2021.

[43] Xiaofeng Qi and Francesco Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. part i: Basic properties of selection and mutation. *IEEE Transactions on Neural Networks*, 5(1):102–119, 1994.

[44] Dirk Thierens. Adaptive mutation rate control schemes in genetic algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pages 980–985. IEEE, 2002.

[45] Tzung-Pei Hong, Hong-Shung Wang, and Wei-Chou Chen. Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of heuristics*, 6:439–455, 2000.

[46] Anant J Umbarkar and Pranali D Sheth. Crossover operators in genetic algorithms: a review. *ICTACT journal on soft computing*, 6(1), 2015.

[47] Shin Ando and Hitoshi Iba. Inference of gene regulatory model by genetic algorithms. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 712–719. IEEE, 2001.

[48] Nasimul Noman, Taku Monjo, Pablo Moscato, and Hitoshi Iba. Evolving robust gene regulatory networks. *PloS one*, 10(1):e0116258, 2015.

[49] Mohammad Al Boni, Derek T Anderson, and Roger L King. Constraints preserving genetic algorithm for learning fuzzy measures with an application to ontology matching. In *Advance Trends in Soft Computing: Proceedings of WCSC 2013, December 16-18, San Antonio, Texas, USA*, pages 93–103. Springer, 2014.

[50] Al Globus, Sean Atsatt, John Lawton, and Todd Wipke. Javagenes: Evolving graphs with crossover. 2000.

[51] Samuel SY Wong and Keith CC Chan. Evoarch: An evolutionary algorithm for architectural layout design. *Computer-Aided Design*, 41(9):649–667, 2009.

[52] Francisco B Pereira, Penousal Machado, Ernesto Costa, and Amílcar Cardoso. Graph based crossover–a case study with the busy beaver problem. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*, pages 1149–1155, 1999.

[53] Anupriya Shukla, Hari Mohan Pandey, and Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm. In *2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)*, pages 515–519. IEEE, 2015.

[54] Noraini Mohd Razali, John Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong, China, 2011.

[55] Carlos M Fonseca and Peter J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1–16, 1995.

[56] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[57] Biao Luo, Jinhua Zheng, Jiongliang Xie, and Jun Wu. Dynamic crowding distance? a new diversity maintenance strategy for moeas. In *2008 Fourth International Conference on Natural Computation*, volume 1, pages 580–585. IEEE, 2008.

[58] Joel Lehman, Kenneth O Stanley, et al. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[59] Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218, 2011.

[60] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, page 40, 2016.

[61] Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

[62] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[63] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM computing surveys (CSUR)*, 33(3):273–321, 2001.

[64] Keinosuke Fukunaga and David R Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, 100(2):176–183, 1971.

[65] Amir Massoud Farahmand, Csaba Szepesvári, and Jean-Yves Audibert. Manifold-adaptive dimension estimation. In *Proceedings of the 24th international conference on Machine learning*, pages 265–272, 2007.

[66] Nikolaos Kouiroukidis and Georgios Evangelidis. The effects of dimensionality curse in high dimensional knn search. In *2011 15th Panhellenic Conference on Informatics*, pages 41–45. IEEE, 2011.

[67] Peter N Yianilos. Data structures and algorithms for nearest neighbor. In *Proceedings of the ACM-SIAM Symposium on Discrete algorithms*, volume 66, page 311, 1993.

[68] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Vldb*, volume 97, pages 426–435, 1997.

[69] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104, 2006.

[70] Dominik Maria Endres and Johannes E Schindelin. A new metric for probability distributions. *IEEE Transactions on Information theory*, 49(7):1858–1860, 2003.

[71] Ivo Grosse, Pedro Bernaola-Galván, Pedro Carpena, Ramón Román-Roldán, Jose Oliver, and H Eugene Stanley. Analysis of symbolic sequences using the jensen-shannon divergence. *Physical Review E*, 65(4):041905, 2002.

[72] Victor M Panaretos and Yoav Zemel. Statistical aspects of wasserstein distances. *Annual review of statistics and its application*, 6:405–431, 2019.

[73] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99, 2000.

[74] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *Advances in neural information processing systems*, 32, 2019.

[75] Alvaro Sanchez, Hernan G Garcia, Daniel Jones, Rob Phillips, and Jané Kondev. Effect of promoter architecture on the cell-to-cell variability in gene expression. *PLoS computational biology*, 7(3):e1001100, 2011.

[76] David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 296–303, 1996.

[77] Awad H Al-Mohy and Nicholas J Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3):970–989, 2010.

[78] Flávio K Miyazawa, Phablo FS Moura, Matheus J Ota, and Yoshiko Wakabayashi. Partitioning a graph into balanced connected classes: Formulations, separation and experiments. *European Journal of Operational Research*, 293(3):826–836, 2021.

[79] Janka Chlebíková. Approximating the maximally balanced connected partition problem in graphs. *Information Processing Letters*, 60(5):225–230, 1996.