EPiC Series in Computing

Volume 70, 2020, Pages 149–162

Proceedings of the 12th International Conference on Bioinformatics and Computational Biology

Multi-objective Optimisation of Gene Regulatory Networks: Insights from a Boolean Circadian Clock Model

Ozgur E. Akman[1]* and Jonathan E. Fieldsend[2]

[1] Department of Mathematics, University of Exeter, Exeter, United Kingdom
O.E.Akman@exeter.ac.uk

[2] Department of Computer Science, University of Exeter, Exeter, United Kingdom
J.E.Fieldsend@exeter.ac.uk

Abstract

The gene regulatory networks that comprise circadian clocks modulate biological function across a range of scales, from gene expression to performance and adaptive behaviour. These timekeepers function by generating endogenous rhythms that can be entrained to the external 24-hour day-night cycle, enabling organisms to optimally time biochemical processes relative to dawn and dusk. In recent years, computational models based on differential equations, and more recently on Boolean logic, have become useful tools for dissecting and quantifying the complex regulatory relationships underlying the clock's oscillatory dynamics. Optimising the parameters of these models to experimental data is, however, non-trivial. The search space is continuous and increases exponentially with system size, prohibiting exhaustive search procedures, which are often emulated instead via grid-searching or random explorations of parameter space. Furthermore, to simplify the search procedure, objective functions representing fits to individual experimental datasets are often aggregated, meaning the information contained within them is not fully utilised.

Here, we examine casting this problem as a *multi-objective* one, and illustrate how the use of an evolutionary optimisation algorithm — the multi-objective evolution strategy (MOES) — can significantly accelerate the parameter search procedure. As a test case, we consider an exemplar circadian clock model based on Boolean delay equations — dynamic models that are discrete in state but continuous in time. The discrete nature of the model enables us to directly compare the performance of our optimiser to grid searches based on enumeration of the parameter space at a fixed resolution. We find that the MOES generates near-optimal parameterisations in computation times which are several orders of magnitude faster than the grid search. As part of this investigation, we also show that there is a distinct trade-off between the performance of the clock circuit in free-running and entrained photic environments. Importantly, runtime results indicate that the use of multi-objective evolutionary optimisation algorithms will make the investigation of larger and more complex models computationally tractable.

*Corresponding author

Q. Ding, O. Eulenstein and H. Al-Mubaid (eds.), BICOB 2020 (EPiC Series in Computing, vol. 70), pp. 149–162

1 Introduction

Mathematical modelling of gene regulatory networks (GRNs) is a critical component of computational biology. Arguably, parameter optimisation is the key bottleneck in constructing such models, because the kinetic parameters quantifying the biochemical processes comprising a GRN are typically unknown — or difficult to measure *in vivo* — and are therefore constrained by fitting to experimental data [1–4]. Comprehensive parameter space explorations are necessary for alternative models of a GRN to be quantitatively compared and for the assumptions made in constructing a model to be rigorously assessed [5–10]. Unfortunately, comprehensive searches rapidly become intractable with increasing model size due to the parameter explosion problem. Currently, this issue restricts the complexity and predictive power of the GRN models that can be constructed. Consequently, there is a pressing need for robust optimisation algorithms capable of dealing with highly parametrised computational biology models, allied with alternative modelling methods that reduce the complexity of the models themselves [11, 12].

In terms of alternative modelling methods, approaches based on Boolean logic provide a significant reduction in complexity compared to more biochemically detailed methods (*e.g.* differential equation modelling). In Boolean models, the activity of each gene is described by a two-state variable taking the value ON (1) or OFF (0), indicating that its products are present or absent, respectively. Biochemical interactions are represented by simple binary functions, or *logic gates*, that calculate the state of a gene from the activation state of its upstream components [13–17]. This approximation dramatically reduces both the number of system components and the number of system parameters, whilst still enabling complex dynamical behaviour to be reproduced [4]. In terms of optimisation algorithms, computational biology has not yet fully exploited the powerful techniques afforded by the evolutionary computation field. Moreover, although multi-objective methods have begun to be utilised [18, 19], it is still common to cast the problem as a uni-objective one, in which the objective function is a weighted sum of least-squares terms, each of which measures the goodness of fit of the model to a dataset measured under a particular experimental protocol (*e.g.* temperature, light environment, genetic background *etc.*) [1–4, 20, 21]. However, combining multiple criteria into a single measure to optimise requires *a priori* expression of preferences, and if a linear sum is used, and the criteria trade-off is non-convex, then only extreme solutions will actually be returned (see *e.g.* [22]).

Here, we demonstrate that a better approach is to cast the problem as a multi-objective one. As a test case, we optimise an exemplar Boolean GRN model to synthetic experimental data, comparing our results with those obtained previously using grid-searching [4]. The multi-objective optimiser is seen to provide solutions equivalent to or better than those achieved through grid search in orders of magnitude fewer function evaluations. It also does not require *a priori* preference articulation in terms of criteria, which may bias the final solution returned, instead returning a set of optimal trade-off solutions to be selected from *a posteriori*.

2 Modelling circadian clocks

Circadian clocks are gene networks found in almost all organisms, controlling biological processes ranging from cyanobacterial cell division to human sleep-wake cycles [23]. These networks operate by generating endogenous ∼24 hour oscillations in gene expression that can synchronise to the external light-dark cycle. This *entrainment* process enables organisms to optimally time biochemical processes relative to dawn and dusk, thereby providing an adaptive advantage [24]. The core clocks of different organisms appear to have a similar structure, based on interlocked sets of negative gene-protein feedback loops augmented by additional positive loops [25].

2.1 An exemplar clock model

Computational models of these feedback structures have proved useful in elucidating the general design principles of clocks, and have also led to the discovery of novel circadian regulators [1–4, 21, 25–28]. The majority of clock models constructed thus far have been based on differential equations and possess on the order of 10 to 100 parameters [4, 25]. A canonical model of this type is the minimal ordinary differential equation (ODE) model of the clock in the fungus *Neurospora crassa*, based on a single negative feedback loop in which the *frequency* (*FRQ*) gene is repressed by its protein product. *FRQ* transcription is upregulated by light, thereby providing a mechanism for light entrainment [29]. The model comprises 3 differential equations describing the dynamics of *FRQ* mRNA and the cytoplasmic and nuclear forms of FRQ protein:

$$\begin{aligned}
\frac{dM}{dt} &= (v_s + L(t))\frac{k_I^N}{k_I^N + P_n^N} - v_m\frac{M}{k_m + M},\\
\frac{dP_c}{dt} &= k_s M - v_d\frac{P_c}{k_d + P_c} - k_1 P_c + k_2 P_n,\\
\frac{dP_n}{dt} &= k_1 P_c - k_2 P_n,\\
L(t) &= \begin{cases} L_M \text{ if } t_{DAWN} \leq \mathrm{mod}(t, 24) \leq t_{DUSK},\\ 0 \text{ otherwise.}\end{cases}
\end{aligned} \tag{1}$$

A circuit diagram of the model can be seen in Fig. 1(a). As is common for models of this type, Hill and Michaelis-Menten kinetics are assumed for transcription and degradation respectively, whilst translation and transport are modelled as first order reactions. Collectively, the reactions are parameterised by 10 kinetic constants, the values of which are taken from the original study [29]. In the M equation of system (1), the forcing term $L(t)$ models the effect of light, which perturbs the clock by upregulating *FRQ* transcription. Setting L to 0 simulates constant darkness (DD), yielding free-running oscillations with a period of just under 24 hours [29]. Entrainment to light-dark (LD) cycles is modelled by switching L between 0 and a maximum value L_M at lights-on (t_{DAWN}), and then switching L back to 0 at lights-off (t_{DUSK}).

2.2 The Boolean model formalism

In this study, we consider a particular class of Boolean models that are well-suited to simulations of GRNs — Boolean delay equations (BDEs) [4, 14, 16, 17, 30]. BDEs are parameterised by a collection of logic gates together with the set of signalling delays that specify the time it takes for state changes to take effect [14]. In previous work, Akman *et al.* introduced a general scheme for modelling biochemical oscillators using this formalism, constructing Boolean versions of several clock models [4]. These included model (1), which can be expressed in general BDE form as:

$$x_M(t) = G(x_P(t-\tau_2), g_2)\,\mathrm{OR}\,L(t-\tau_3), \quad x_P(t) = G(x_M(t-\tau_1), g_1). \tag{2}$$

The circuit diagram for this model is shown in Fig. 1(b). In (2), x_M is *FRQ* mRNA, x_P is lumped FRQ protein (combining the cytoplasmic and nuclear forms) and $\boldsymbol{\tau} = (\tau_1, \tau_2, \tau_3)$ are the signalling delays. The light input $L(t)$ is as in eqn. (1), with L_M taking values 0 or 1.

The function $G(x, g)$ in (2) implements the identity or NOT gate, modelling activation and repression by x respectively, depending on the value of the bit g: $G(x, 0) = x, G(x, 1) = \mathrm{NOT}\ x$. The bitstring $\mathbf{g} = g_1 g_2$ thus specifies the collection of logic gates, referred to as the *logic configuration (LC)*. There are 4 possible LCs, consistent with the underlying directed graph of

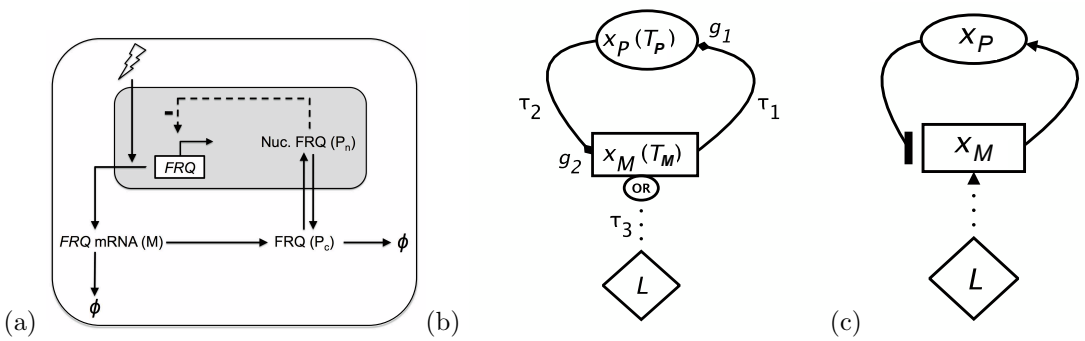

Figure 1: **(a)** Circuit diagram of the ODE model (1) for the *N. crassa* circadian clock. *FRQ* mRNA (M) is translated into protein (P_c) in the cytoplasm and then transported into the nucleus (P_n) where it represses *FRQ* transcription. Light (flash symbol) entrains the model by upregulating transcription. **(b)** Circuit diagram of the general Boolean formulation (2) of (a). x_M denotes *FRQ* mRNA and x_P denotes bulk FRQ protein. L indicates light, $\{\tau_1, \tau_2, \tau_3\}$ represent the signalling delays and $\{T_M, T_P\}$ are the discretisation thresholds used to fit the discrete model to continuous data. $g_1, g_2 \in \{0, 1\}$ index logic gates that can be varied to generate different regulatory structures, or logic configurations (LCs). The LC consistent with the ODE model (the DE LC) is $\{g_1 = 0, g_2 = 1\}$ — its circuit diagram is plotted in **(c)**.

the ODE model (*i.e.* the graph with connections $M \to P_c \to P_n \to M$), of which only one, $\mathbf{g} = 01$, is consistent with the pattern of activation/inhibition of the ODE model (activation of x_P by x_M; repression of x_M by x_P – *cf.* Fig. 1(c)). This configuration is hence referred to as the *DE LC*. The other 3 configurations represent alternative regulatory structures consistent with the model graph: *e.g.* $\mathbf{g} = 00$ corresponds to a double positive feedback loop circuit.

3 Optimisation protocol

3.1 Synthetic data & thresholding

Following the approach used in [4], we optimise the BDE model (2) to synthetic mRNA and protein data generated from the corresponding ODE formulation (1) using the variant of Gillespie's stochastic simulation algorithm introduced by Gonze *et al.* [31]. 4 independent sets of $\{M(t), P_c(t), P_n(t)\}$ timeseries were generated over the interval $0 \leq t \leq 120$ (5 circadian cycles), starting from an initial condition on the system attractor; next, the cytoplasmic and nuclear proteins were summed to obtain a bulk protein variable $P = P_c + P_n$; finally the mRNA and protein traces $\{(M(t), P(t)) : 0 \leq t \leq 120\}$ were sampled every $\tau_S = 0.5$h, reflecting the minimum possible experimental resolution [4]. Here, we present the results obtained for one set of timeseries (shown in Figs. 6(a-b)) that is representative of the results obtained with the others.

In order to enable this data to be compared to the solutions of (2) for a given logic configuration $\boldsymbol{g}$ and delay set $\boldsymbol{\tau}$, the sampled timeseries were normalised between 0 and 100, and then discretised by applying thresholds $0 < T_M < 100$ (mRNA) and $0 < T_P < 100$ (protein), such that all subthreshold values are set to 0 (OFF) and all suprathreshold values to 1 (ON). This yields the sampled, discretised synthetic mRNA and bulk protein timeseries $\{(D_M(t; T_M), D_P(t; T_P)) : 0 \leq t \leq 120, t/\tau_S \in \mathbb{Z}\}$. The first cycle of discretised data is then

used to generate predicted timeseries $(\hat{x}_M(t), \hat{x}_P(t))$ from (2) that are scored against the last 4 circadian cycles of data, as described below in sections 3.2 and 3.3.

3.2 Generating model predictions

The model prediction $(\hat{x}_M(t), \hat{x}_P(t))$ is calculated as

$$(\hat{x}_M(t), \hat{x}_P(t)) = \begin{cases} (D_M(t; T_M), D_P(t; T_P)), & \text{if } 0 \leq t \leq 24, \\ (G(P_I(t), g_2) \text{ OR } L(t - \tau_3), G(M_I(t), g_1)), & \text{if } 24 < t \leq 120, \end{cases} \tag{3}$$

where $P_I(t)$ and $M_I(t)$ represent the protein and mRNA timeseries used to generate the prediction in the projection interval $24 < t \leq 120$. We consider two choices for these timeseries that yield different prediction methods. The first uses only the data timeseries for projection, and is obtained by setting $P_I(t) = D_P(t - \tau_2; T_P)$ and $M_I(t) = D_M(t - \tau_1; T_M)$. As the predictions are obtained from each model equation independently of the other one, this method is referred to as *serial updating*. The second method numerically approximates true solutions of (2) by solving the equations simultaneously, with the first cycle of data used as the system history [14]. It is obtained by setting $P_I(t) = \hat{x}_P(t - \tau_2)$ and $M_I(t) = \hat{x}_M(t - \tau_1)$. As the predictions are now dependent on one another, this method is referred to as *parallel updating*.

We note that the thresholds (T_M, T_P) act as meta-parameters in (3): although they do not directly affect the form of model (2), which only depends on the delays $\boldsymbol{\tau}$ and logic gates $\mathbf{g}$, they determine the data used to compute model predictions and to cost predictions against. We further note that for a given combination of delays $\boldsymbol{\tau}$ and thresholds $\mathbf{T}$, an exact prediction generated using parallel updating (*i.e.* one for which $\hat{x}_M(t) \equiv D_M(t; T_M)$ and $\hat{x}_P(t) \equiv D_P(t; T_P)$) is, by definition, an exact solution of (2), with the data as the initial history; furthermore, it will automatically satisfy the serial update equations. However, the converse is not true: an exact prediction $(\hat{x}_M(t), \hat{x}_P(t))$ obtained by serial updating will not necessarily satisfy the parallel update equations. Accurate predictions obtained with serial updating thus generate timeseries that are *consistent* with the model; some of these will also be true solutions of (2). Serial updating can thus be considered a computationally cheap way of generating putative model solutions. Following [4], we use the latter for all optimisations, and then implement parallel updating to select viable models from the resulting ensemble of potential designs. As in [4], both serial and parallel updating were implemented using timestepping, with the timestep Δt set to the sampling interval τ_S. Consequently, all τ_ks are taken as integer multiples of Δt.

3.3 Objective functions

In order to assess the goodness-of-fit of the model predictions $(\hat{x}_M(t), \hat{x}_P(t))$ generated using serial updating, two objective functions $\hat{f}_{DD}$ and $\hat{f}_{LD}$ were used, scoring the model against timeseries simulated in two light regimes: (i) constant darkness (termed DD), obtained by setting $L_M = 0$ in the $L(t)$ equation (*cf.* eqns. (1)); and (ii) 12:12 light-dark (LD) cycles (alternating 12h periods of light and dark), obtained by setting $L_M = 1$, $t_{DAWN} = 6$ and $t_{DUSK} = 18$ in the $L(t)$ equation. The individual objectives $\hat{f}_{DD}$ and $\hat{f}_{LD}$ are calculated as $\hat{f}_i(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}) = \sum_{j=M,P} \hat{f}_{i,j}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T})$, where

$$\hat{f}_{i,j}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}) = \frac{1}{N(\tau_S)} \sum_{k=24/\tau_S}^{120/\tau_S} d_H(\hat{x}_j(k\tau_S; \boldsymbol{\tau}, \mathbf{g}, \mathbf{T}, \mathbf{D}_i(\mathbf{T})), D_{i,j}(k\tau_S; T_j)), \tag{4}$$

and where $\mathbf{D}_i(\mathbf{T}) = \{(D_{i,M}(t;T_M), D_{i,P}(t;T_P)) : 0 \leq t \leq 120)\}$ denotes the discretised data generated in light condition i, $\hat{x}_j(k\tau_S; \boldsymbol{\tau}, \mathbf{g}, \mathbf{T}, \mathbf{D}_i(\mathbf{T}))$ is the corresponding prediction for variable j generated from (3) (with all dependencies written explicitly), $N(\tau_S) = {}^{96}/_{\tau_S} + 1$ is the number of timeseries points used for costing and $d_H(\cdot,\cdot)$ is Hamming distance between bitstrings. The discrepancy $\hat{f}_{i,j}$ between prediction and data for each variable is thus measured as the Hamming distance normalised by bitstring length [4,32]. It follows that $0 \leq \hat{f}_i \leq 2$, with lower costs indicating better fits of the model to data in each case.

In optimising model predictions to synthetic data, it is possible to obtain designs $(\boldsymbol{\tau}, \mathbf{T}, \mathbf{g})$ such that the thresholds are close to 0 or 100. In such cases, the discretised data contains bitstrings composed almost entirely of 0s or 1s. This can yield solutions that, whilst having low objective values, contain little temporal information [4,30]. To mitigate against this problem, we introduce a penalty term for each objective, based on calculating the score that would be obtained by chance from timeseries containing the same proportion of 0s and 1s as the prediction and discretised data [4,30]. Formally, given a threshold combination $\mathbf{T}$, for each light condition i and variable j, let $p_{i,j}$ and $\hat{p}_{i,j}$ denote the proportion of ones in the resulting thresholded data and model prediction, respectively. Further, let $\{\eta^D_{i,j,k}, \eta^M_{i,jk} : 1 \leq k \leq N(\tau_S)\}$ be i.i.d. Bernoulli random variables with means $p_{i,j}$ and $\hat{p}_{i,j}$, respectively. Then, if $\eta^D_{i,j,k}$ and $\eta^M_{i,j,k}$ are independent for all k, the normalised Hamming distance between the random bitstrings is given by $\frac{1}{N}\sum_{k=1}^{N} d_H(\eta^D_{i,j,k}, \eta^M_{i,j,k})$, which follows a Binomial distribution with mean $\mu_{i,j}$ and standard deviation $\sigma_{i,j} = \sqrt{\mu_{i,j}(1-\mu_{i,j})/N}$, where $\mu_{i,j} = p_{i,j} + \hat{p}_{i,j} - 2p_{i,j}\hat{p}_{i,j}$. Thus, as the sampling interval $\tau_S \to 0$, $N(\tau_S) \to \infty$ and hence $\sigma_{i,j} \to 0$; *i.e.*, the distribution of normalised Hamming distances converges to its mean value, $\mu_{i,j}$. The optimisation results obtained previously for the uni-objective optimisation problem returned threshold values T_j that maximised the binary entropy of the data, *i.e.* thresholds yielding $p_{i,j}, \hat{p}_{i,j} \approx {}^1/_2$ [4,30]. For such solutions, $\mu_{i,j} \approx {}^1/_2$. Accordingly, we define the final penalised cost function f_i for $i = \{DD, LD\}$ as

$$f_i(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}) = \sum_{j=M,P} f_{i,j}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}); \quad f_{i,j}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}) = \hat{f}_{i,j}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}) + \frac{1}{2} - \mu_{i,j}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}). \tag{5}$$

This means that for biologically feasible solutions, $f_{i,j} \approx \hat{f}_{i,j}$, and so no penalty is applied. However, extreme thresholds yielding good scores (*i.e.* for which $p_{i,j} \approx 0$ or $p_{i,j} \approx 1$, but $\hat{f}_{i,j} \approx \mu_{i,j}$) will be penalised, receiving a cost score of $\approx {}^1/_2$, rather than 0.

3.4 Multi-objective optimisation

Here, rather than summing f_{DD} and f_{LD} as was done in previous studies [4,30], we instead use a multi-objective approach, in which the objectives are optimised together, with no *a priori* assumptions about the relative importance of each. In multi-objective optimisation, we seek the *Pareto optimal* set of *trade-off* solutions — those where the improvement on any single f_i, by varying a solution, $\mathbf{x}$, within some feasible domain, $\mathcal{X}$, can only be achieved by degrading the performance on one or more other criteria. Selection of a single solution (model) can then be via *a posteriori* preference articulation [33]. Formally, a design $\mathbf{x}$ is said to *dominate* another design $\mathbf{x}'$ ($\mathbf{x} \prec \mathbf{x}'$) if it is no worse on all criteria and strictly better on at least one. The set of all possible Pareto optimal solutions (the *Pareto set*) is defined as $\mathcal{P} = \{\mathbf{x} \in \mathcal{X} | \nexists \mathbf{x}' \prec \mathbf{x}, \mathbf{x}' \in \mathcal{X}\}$. The image of the Pareto set in objective space is known as the *Pareto front*, $\mathcal{F}$.

This definition of dominance allows us to put a partial order on any two solutions (dominated, dominating and mutually non-dominating). Given an exhaustive search over $\mathcal{X}$, this ordering allows the Pareto set to be found (the non-dominated subset). However, often such

exhaustive search is infeasible, and so we must rely on search methods which deliver an approximation to $\mathcal{P}$. Probably the most popular family of such methods are those in nature-inspired computation. These heuristic/stochastic approaches aim to roughly mimic the ways in which various problems are solved in the natural world, via a computational analogue. Examples include genetic algorithms and evolution strategies (evolution), and particle swarm optimisation (cognitive and social aspects of search in swarms/flocks), as well as many others [33–35].

3.5 The optimiser

In order to tackle the optimisation of our BDE model, we implemented a domain-specific *multi-objective evolution strategy (MOES)*, in which the parameter vector is defined as the combination of delays and thresholds, $\mathbf{x} = (\tau_1, \tau_2, \tau_3, T_M, T_P)$, and the objective vector is the combined vector of DD and LD scores, $\mathbf{y} = (f_{DD}, f_{LD})$. The constraints defining $\mathcal{X}$ are

$$0 \leq \tau_1, \tau_2 \leq 23.5, \quad 0 \leq \tau_1 + \tau_2 \leq 23.5, \quad 0 \leq \tau_3 \leq 12, \quad 0 \leq T_M, T_P \leq 100. \tag{6}$$

The delay constraints ensure that the sum of the signalling delays around the negative feedback loop do not sum to a value greater than the period of free-running or entrained oscillations [4,30]. The grid search employed previously in [4] sought to minimise $f_{DD} + f_{LD}$, and exhaustively searched over the finite, *discrete* design space, $\mathcal{X}_{discrete}$, obtained by choosing delay-threshold combinations with $\tau_k = \{0.5p : p \in \mathbb{N} \cup \{0\}\}$ and $T_j = \{2q : q \in \mathbb{N} \cup \{0\}\}$, subject to the constraints defined in (6). This required a total of 67,300,875 calculations of $\mathbf{y}$ for each LC $\mathbf{g}$. The MOES attempts to provide approximately the same results as the exhaustive grid search, but with significantly fewer function evaluations. Briefly, the MOES iteratively evolves an archive, $\mathcal{A}$, comprising an estimate of the Pareto set, $\mathcal{P}$. The algorithm starts by generating m random solutions, from which non-dominated members are extracted to initialise $\mathcal{A}$. At each subsequent iteration, a member of $\mathcal{A}$ is randomly selected for mutation, and if its offspring is not dominated by any elements of $\mathcal{A}$, it is inserted into it, and any members dominated by it are removed. The algorithm continues until n function evaluations have been executed. Detailed pseudocode for the optimiser is given in section A1.1.

3.6 Optimisation experiments

For each fixed choice of gates $\mathbf{g} = g_1 g_2$, we optimise f_{DD} and f_{LD} over the combination of delays $\boldsymbol{\tau} = (\tau_1, \tau_2, \tau_3)$ and thresholds $\mathbf{T} = (T_M, T_P)$, over the following two choices for $\mathcal{X}$: (i) the discrete space $\mathcal{X}_{discrete}$ defined above, in which the delays τ_k are integer multiples of the sampling interval τ_S and the thresholds are varied in steps of 2%; and (ii) a *mixed integer* space, $\mathcal{X}_{mixed}$, in which the delays are varied as in (i), but thresholds vary continuously. As $\mathcal{X}_{discrete}$ was fully explored in the previous uni-objective optimisation problem considered in [4], experiment (i) enables us to directly compare the performance of our optimiser against grid search on the multi-objective version of the problem, executed on the same domain. Experiment (ii) assesses whether the optimiser is able to find *superior* solutions to those obtained with grid search, when the domain is no longer discrete. In both experiments, a key aim is to assess the ability of the optimisation protocol to recover the DE LC $\mathbf{g} = 01$ — the configuration corresponding to the ODE model used to generate the synthetic fitting data (see Fig. 1(c)).

3.7 Assessing optimiser performance

The set of solutions found using the MOES should be well converged to the Pareto front. Furthermore, given that we are using a stochastic search process, we are concerned that the

algorithm should perform consistently. A popular measure used to test solutions for these properties is the *hypervolume* measure [36]. This calculates the proportion of volume in objective space that is dominated by the Pareto front $\mathcal{F}$, which is also dominated by the approximation to $\mathcal{F}$ generated by the optimiser. Note that in order to calculate this, we must have access to $\mathcal{F}$. For $\mathcal{X}_{discrete}$, the results from the exhaustive grid search give us this set, which we denote $\mathcal{F}_{discrete}$. A *reference point* also needs to be defined for the hypervolume calculation (which, along with the Pareto front, defines the bounds of the volume being assessed). Here, we know the worst possible objective values $\{f_{DD} = 2, f_{LD} = 2\}$, and so a natural reference point is (2,2). Given these settings, if $\mathcal{H}(\mathcal{A})$ is high, this indicates that the members of $\mathcal{A}$ are well converged to $\mathcal{P}_{discrete}$ and when $\mathcal{H}(\mathcal{A}) = 1$ then $\mathcal{A} = \mathcal{P}_{discrete}$ (full convergence of the approximation to the true front). For $\mathcal{X}_{mixed}$, we do not have access to the true front, as the full search space is computationally intractable. However, we can still compute hypervolume using $\mathcal{F}_{discrete}$ and the worst-case reference point, with $\mathcal{H}(\mathcal{A})$ values *greater* than 1 now taken to indicate that the mixed integer optimiser can find solutions *better* than those possible under the exhaustive grid-search of $\mathcal{X}_{discrete}$.

4 Results

The optimiser described above was run for a total of $n = 100,000$ function evaluations on 100 separate runs (initialised from $m = 100$ uniformly drawn parameterisations — a different set for each run). Visualising the cost landscape searched by the optimiser is non-trivial. We can, however, illustrate the density mapping of solutions by uniformly sampling from $\mathcal{X}$, and plotting the corresponding points $(f_{DD}(\mathbf{x}), f_{LD}(\mathbf{x}))$ in each case. This is illustrated in Fig. 2 (top row), where 1,000,000 uniform samples from $\mathcal{X}$ are taken. Across gate combinations, we clearly see that a high density of solutions are mapped to the region around $(f_{DD}, f_{LD}) = (1, 1)$ — models that effectively behave at random. Also plotted are the members of $\mathcal{F}_{discrete}$ — the Pareto front found by exhaustive grid-search of $\mathcal{X}_{discrete}$. As can be seen, these are clearly drawn from very small regions of $\mathcal{X}_{discrete}$, indicating that the set of target points for the optimiser lies in a region of the search space that is extremely difficult to locate using random search alone.

4.1 Searching over the discrete parameter space

For the first set of experiments, the random initialisation was drawn from the discrete delay-threshold parameter space, $\mathcal{X}_{discrete}$, and the discretised Gaussians used in the MOES variation operations were configured such that only $\mathbf{x}$ combinations lying in $\mathcal{X}_{discrete}$ could be proposed for evaluation. This meant that we had access to the actual Pareto set for this optimiser configuration ($\mathcal{P}_{discrete}$), and could assess convergence rates (and consistency) to the corresponding Pareto front ($\mathcal{F}_{discrete}$). Fig. 3 (top row) illustrates the convergence properties of the MOES in this case. For all logic configurations, the optimiser finds a reasonable approximation to $\mathcal{F}_{discrete}$ within 1,000 function evaluations (on average to within 2% of $\mathcal{F}_{discrete}$), and by 10,000 function evaluations, the majority of optimiser runs have found $\mathcal{F}_{discrete}$. This is in contrast to a uniform random search of $\mathcal{X}_{discrete}$ (also plotted) which after 100,000 function evaluations has still not reached the same level that the MOES achieved in just 1,000 evaluations. The figure also shows the consistency over time of optimiser performance, via the rapid tightening of the interquartile ranges to the median hypervolume.

The progression of the optimiser over time can be more directly visualised by examining the distributions plotted from a single run of the MOES shown in Fig. 2 (bottom row). For each of the gates, the leading edge of the distribution (lower left edge) exhibits convergence to

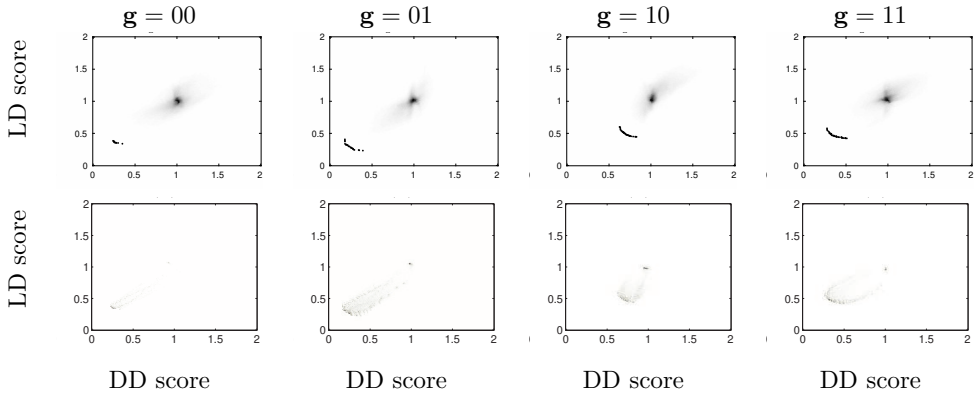

Figure 2: Cost distributions for each logic configuration $\mathbf{g} \in \{0,1\}^2$. Top row: approximation by uniformly sampling 1,000,000 points in $\mathcal{X}$ and plotting the corresponding values of f_{DD} and f_{LD} as biobjective grid densities. The Pareto front $\mathcal{F}_{discrete}$ obtained by exhaustive grid search is also shown (collection of darker points lying in lower left region of plot in each case). Bottom row: 100,000 function evaluations during a single run of the discrete space optimiser.

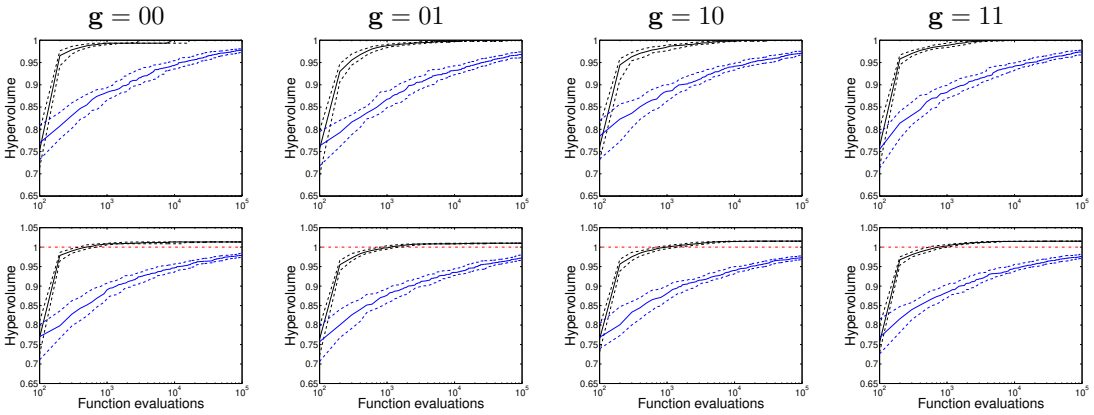

Figure 3: MOES convergence, as quantified by the hypervolume indicator, $\mathcal{H}(\mathcal{A})$. In each plot, the solid black line denotes the median hypervolume (over 100 runs) of the estimate stored by the MOES as the search progresses and the solid blue line shows the result obtained via uniform sampling (note the abscissa is on a log scale). Dashed lines show the 25th and 75th percentile results. Top row: convergence of the discrete space optimiser to $\mathcal{F}_{discrete}$. An $\mathcal{H}(\mathcal{A})$ value of 1 indicates the Pareto front is attained. Bottom row: convergence of the mixed integer space optimiser, with hypervolume computed as for the discrete search. Here $\mathcal{F}_{mixed}$ is unknown and an $\mathcal{H}(\mathcal{A})$ value greater than 1 indicates that the mixed integer optimiser is able to find solutions *better* than those in $\mathcal{F}_{discrete}$. This is marked with a dot-dashed (red) line for convenience.

$\mathcal{F}_{discrete}$, but a distinct trailing edge of less fit solutions can be seen leading away from this area. However, comparison with the random search (top row) clearly demonstrates that the optimiser rapidly converges towards the region of $\mathcal{X}_{discrete}$ containing the solutions of highest quality (in particular, note the comparatively low density of points around the (1,1) random

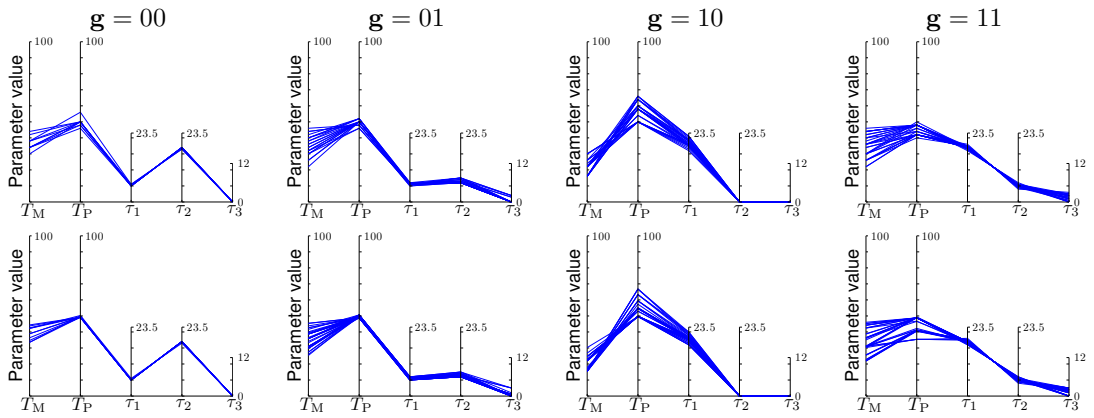

Figure 4: Parallel coordinate plots of the parameter space locations of the estimated Pareto sets from the median performing run of each optimisation experiment. Top row: discrete parameter search. Bottom row: mixed integer parameter search.

performance region). Fig. 4 (top row) shows parallel coordinate plots of the parameter space locations of the archive of the median performing runs for each logic configuration.

4.2 Searching over the mixed integer parameter space

For the next experiments, the delay parameters $\boldsymbol{\tau}$ were initialised and varied to discrete values as before; however the thresholds $\mathbf{T}$ were initialised in the continuous space (enclosed by the maximum and minimum permitted values) and the MOES variation operator for thresholds was not discretised as solutions were evolved. This meant the search space became considerably larger, but allowed us to investigate if solutions might exist in this space, $\mathcal{X}_{mixed}$, which outperformed those in $\mathcal{X}_{discrete}$. In particular, when we are optimising across $\mathcal{X}_{mixed}$, we do not have an $\mathcal{F}_{mixed}$ to compare to. However, we can use $\mathcal{F}_{discrete}$ as a baseline to reach/improve upon. Writing $\mathbf{y} = (f_{DD}(\mathbf{x}), f_{LD}(\mathbf{x}))$ for $\mathbf{x} \in \mathcal{X}$, we note that since $\mathcal{X}_{discrete} \subset \mathcal{X}_{mixed}$, then $\mathbf{y}_{mixed} \preceq \mathbf{y}_{discrete} | \mathbf{y}_{mixed} \in \mathcal{F}_{mixed}, \mathbf{y}_{discrete} \in \mathcal{F}_{discrete}$. Results are provided in Fig. 3 (bottom row). As can be seen, by 10,000 function evaluations this wider search has already found solutions equivalent in quality to those obtained with exhaustive grid-searching over $\mathcal{X}_{discrete}$, and subsequently finds solutions that are even better, as quantified by hypervolume values $\mathcal{H}(\mathcal{A})$ with $\mathcal{H}(\mathcal{A}) > 1$. Fig. 4 (bottom row) shows parallel coordinate plots for the archive of median performing runs for each logic configuration. The parameters are located in similar ranges to those found by the discrete search (*cf.* the plots in the top row of Fig. 4); however some can be seen to lie in narrower bounded regions, which the continuous thresholds have rendered legal.

4.3 Comparing the discrete and mixed integer parameter searches

Fig. 5 (left plots) compares $\mathcal{F}_{discrete}$ with the estimates returned by the two implementations of the MOES. The median performing result (w.r.t. hypervolume) of the corresponding 100 runs is plotted. For each logic configuration (LC), the grid-search utilised 67,300,875 function evaluations; for each search mode (discrete and mixed integer) the MOES utilised 100,000. It can be see that although the grid-search employed 673 times more function evaluations, the discrete space MOES generates excellent approximations to it, with the entirety of $\mathcal{F}_{discrete}$

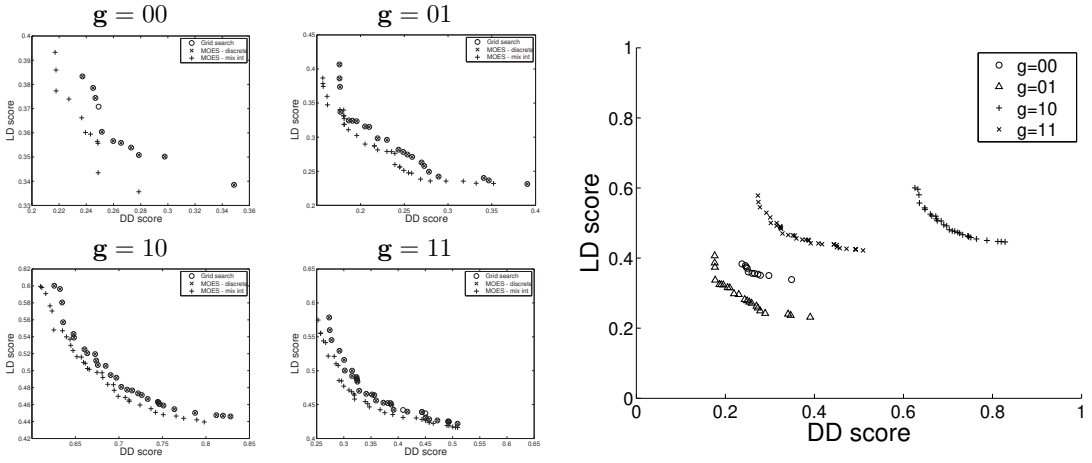

Figure 5: Left plots: comparison of the Pareto fronts computed for each logic combination (LC) using exhaustive grid-search ($\circ$), the MOES searching over the equivalent discrete representation $\mathcal{X}_{discrete}$ ($\times$) and the MOES searching over the mixed integer representation $\mathcal{X}_{mixed}$ ($+$). Right plot: locations of the Pareto front $\mathcal{F}_{discrete}$ in the discrete search space for each LC.

found for LCs $\mathbf{g} = 01$ (the data-generating, DE LC) and $\mathbf{g} = 10$, and only one or two Pareto set members missing for the other two LCs. The results also show that better solutions can be found in $\mathcal{X}_{mixed}$ when compared to $\mathcal{X}_{discrete}$, as the continuous space MOES has located delay-threshold combinations which dominate (in the same number of function evaluations) all of $\mathcal{P}_{discrete}$ for LCs $\mathbf{g} = 00$, 10 and 11. In the case of $\mathbf{g} = 01$, a small portion of the approximation to $\mathcal{F}_{mixed}$ lies behind an element of $\mathcal{F}_{discrete}$. This indicates that there are still better solutions to be found by this particular optimiser run, as $\mathcal{X}_{discrete} \subset \mathcal{X}_{mixed}$. However, discovering $\mathcal{P}_{mixed}$ via exhaustive search is infeasible, so we cannot say how well converged the MOES approximations to it are, although Fig. 3 (bottom row) shows that by 100,000 function evaluations, the algorithm is performing consistently.

The individual Pareto fronts $\mathcal{F}_{discrete}$ obtained for each LC can also be seen in Fig. 5 (right plot), where they are all plotted together. Interestingly, there is a total order on these fronts, with the front for the DE LC $\mathbf{g} = 01$ dominating all the others. Finally, Figs. 6(c-d) shows all the solutions obtained from the Pareto set corresponding to this front, computed with parallel updating in each case (*i.e.* true solutions of eqns. (2)). It can be seen that whilst the periods of the oscillations are quite consistent, the phases at which the model components switch between OFF (0) and ON (1) vary across the front. The MOES has thus returned an ensemble of viable solutions, from which one or more final designs could be selected for further investigation (*e.g.* assessing the predictive performance of the model on an out-of-sample dataset).

5 Discussion & Conclusions

Recent years have seen increasing interest in methods for both reducing the complexity of GRN models and efficiently searching the resulting cost landscapes [12]. However, the majority of work in this area combines grid search (or random search) with local optimisation (*e.g.* simulated annealing or further grid searches at successively finer resolutions) in order to determine the best-fit model [1–4]. A problem with this approach is the computational cost of the prelim-

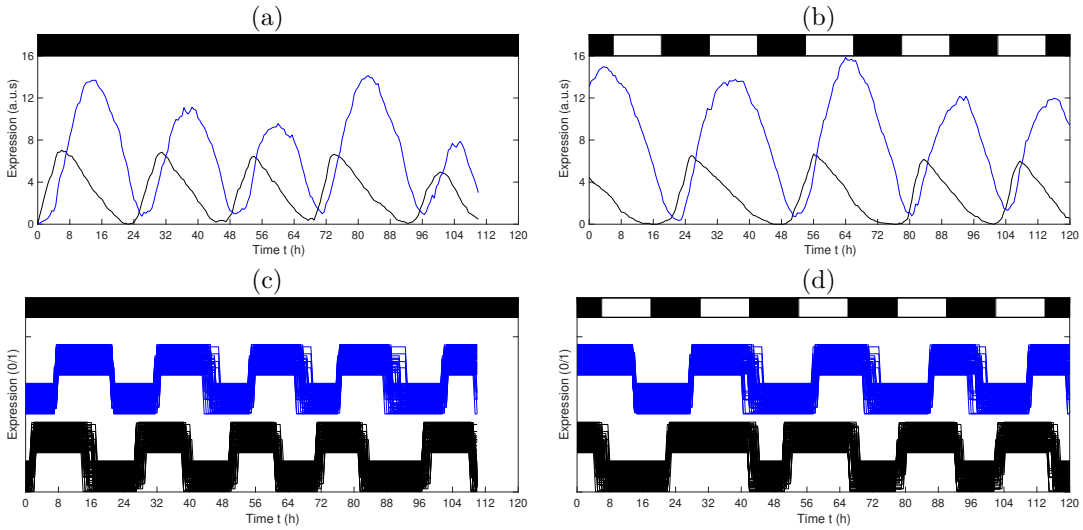

Figure 6: Top row: synthetic mRNA timeseries $M(t)$ (black lines) and protein timeseries $P(t)$ (blue lines) generated from the ODE eqns. (1) in DD (a) and 12:12 LD cycles (b). Simulations of the BDE eqns. (2) in DD (c) and 12:12 LD cycles (d) for all parameter combinations within the Pareto set $\mathcal{P}_{discrete}$ obtained by optimising the model to the synthetic data with the discrete MOES for $\mathbf{g} = 01$. Black/blue lines represent Boolean mRNA/protein timeseries $x_M(t)/x_P(t)$, respectively — these have been offset for clarity, with a more graded offset applied to solutions obtained for different parameter sets. White/black bars represent light/dark, respectively

inary search, which precludes the tackling of anything other than relatively simple problems.

In this study, we have demonstrated that for an exemplar GRN model based on Boolean logic, a multi-objective evolutionary optimiser is able to replicate the performance of grid-searching with 3 orders of magnitude fewer function evaluations. Furthermore, when the constraints on the parameter space were relaxed so that the search became mixed integer rather than discrete, the optimiser was able to find parameter combinations with superior performance to grid-searching, whilst maintaining the same reduction in computational burden. In addition, by utilising the hypervolume indicator, we demonstrated that our optimisation algorithm is robust to both the initial population distribution and the stochasticity of the optimiser.

More generally, these results provide a firm basis for future work on multi-objective optimisation of GRN models, particularly those where different potential architectures can be parametrised. Here, we ran separate optimisers for each logic configuration, obtaining Pareto fronts with a total ordering. However, this may not be the case for other models, where the Pareto fronts for different configurations may overlap, and hence — depending on the desired trade-off — a different architecture may be preferred. As such, optimising across configurations is a useful extension. This can be implemented for Boolean models by simply augmenting the parameter vector with the bits that specify the logic gates, and this simple approach has been shown to work well for the uni-objective problem [30]. It would also be instructive to further explore the use of serial updating as a computationally cheap surrogate for parallel updating. For example, rather than only applying parallel updating at the end of the optimisation procedure, the surrogate error could instead be actively learned by intermittently querying the cost function obtained with parallel rather than serial updating (*e.g.* every other generation).

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council [grant numbers EP/N017846/1 and EP/N014391/1]. We would like to acknowledge the use of the University of Exeter High-Performance Computing (HPC) facility in carrying out this work.

References

[1] J. C. W. Locke *et. al. Mol. Syst. Biol.*, 2:59, 2006.
[2] O. E. Akman *et. al. Mol. Syst. Biol.*, 4:64, 2008.
[3] O. E. Akman, D. A. Rand, P. E. Brown and A. J. Millar. *BMC Syst. Biol.*, 4:88, 2010.
[4] O. E. Akman *et. al. J. Roy. Soc. Interface*, 9(74):2365–2382, 2012.
[5] J. B. Johnson and K. S. Omland. *Trends Ecol. Evol.*, 19(2):101–108, 2004.
[6] M. Ashyraliyev *et. al. FEBS J.*, 276(4):886–902, 2009.
[7] G. Cedersund and J. Roll. *FEBS J.*, 276(4):903–922, 2009.
[8] D. F. Slezak *et. al. PLoS One*, 5(10):e13283, 2010.
[9] G. Lillacci and M. Khammash. *PLoS Comput. Biol.*, 6(3):e1000696, 2010.
[10] L. Mikeev and V. Wolf. *Proc. ACM International Conference on Hybrid Systems*, 155-66, 2012.
[11] R. Adams *et. al. Bioinformatics*, 29(5):664–665, 2013.
[12] I. T. Tokuda, O. E. Akman and J. C. W. Locke. *J. Theor. Biol.*, 463:155–166, 2019.
[13] S. A. Kauffman. *J. Theor. Biol.*, 22(3):437–467, 1969.
[14] D. P. Dee and M. Ghil. *SIAM J. Appl. Math.*, 44(1):111–126, 1984.
[15] R. Thomas. *J. Theor. Biol.*, 153:1–23, 1991.
[16] H. Öktem, R. Pearson and K. Egiazarian. *Chaos*, 13(4):1167–1174, 2003.
[17] M. Ghil, I. Zaliapin and B. Coluzzi. *Physica D*, 237(23):2967–2986, 2008.
[18] R. Gupta *et. al. BMC Syst. Biol.*, 5:52, 2011.
[19] I. Otero-Muras and J. R. Banga. *BMC Syst. Biol.*, 8:113, 2014.
[20] J. R. Banga. *BMC Syst. Biol.*, 2:47, 2008.
[21] A. Pokhilko *et. al. Mol. Syst. Biol.*, 8(1), 2012.
[22] M. Ehrgott. *Multicriteria Optimization*. Springer, 2nd edition, 2005.
[23] J. C. Dunlap, J. J. Loros and P. J. DeCoursey. *Chronobiology*. Sinauer, 2003.
[24] V. Resco, J. Hartwell and A. Hall. *Ecol. Lett.*, 12(6):583–592, 2009.
[25] E. E. Zhang and S. A. Kay. *Nat. Rev. Mol. Cell Biol.*, 11:764–776, 2010.
[26] K. Fogelmark and C. Troein. *PLoS Comput. Biol.*, 10(7):e1003705, 2014.
[27] J. DeCaluwe *et. al. Front. Plant Sci.*, 7(74), 2016.
[28] M. Foo, D.E. Somers and P.-J. Kim. *PLoS Comput. Biol*, 12(2):e1004748, 2016.
[29] J. C. Leloup, D. Gonze, and A. Goldbeter. *J. Biol. Rhythms*, 14(6):433–448, 1999.
[30] K. Doherty, K. Alyahya, O. E. Akman and J. E. Fieldsend. *Proc. GECCO 2017*, 1644–1651, 2017.
[31] D. Gonze, J. Halloy and P. Gaspard. *J. Chem. Phys.*, 116:10997–11010, 2002.
[32] S.E. Harris, B.K. Sawhill, A. Wuensche and S. Kauffman. *Complexity*, 7(4):23–40, 2002.
[33] C. A. Coello Coello, G. B. Lamont and D. A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 5th edition, 2007.
[34] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 1st edition, 2001.
[35] R. Purshouse *et. al. Evolutionary Multi-Criterion Optimization*. LNCS. Springer, 2013.
[36] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.

A Appendix

A1.1 Pseudocode for the multi-objective evolution strategy

The main loop is described in Algorithm 1. The variation mechanism is outlined in Algorithms 2–4. At a high level, as shown in Algorithm 2, after copying the sample (lines 2-3), the copy has a 50% probability of both the delay and threshold elements being perturbed, a 25% probability of *only* varying the delay elements and a 25% probability of *only* varying the threshold elements (lines 4-12). Variation/perturbation itself is accomplished via draws $\mathcal{G}(x, \sigma^2)$ from Gaussian distributions with centre x and variance σ^2, where σ is the square root of the valid range of the variable. Discretised Gaussians are used for threshold perturbations when searching across $\mathcal{X}_{discrete}$, and for delay perturbations when searching across both $\mathcal{X}_{discrete}$ and $\mathcal{X}_{mixed}$. As outlined in Algorithm 3, when the thresholds $\mathbf{T}$ are varied, there is an equal probability of both thresholds being varied, only the first threshold being varied and only the second threshold being varied. Line 12 ensures that movements *outside* of $\mathcal{X}$ are truncated to the legal limit.

Algorithm 1 The multi-objective optimiser. m is the number of initial random solutions, n is the total number of evaluations and $\mathcal{D}$ is the data.

1: **procedure** OPTIMISE($m, n, \mathcal{D}, \mathbf{x}^{max}, \mathbf{x}^{min}$)
2: $X :=$ `initialise`$(m, \mathbf{x}^{max}, \mathbf{x}^{min})$
3: $Y :=$ `evaluate`$(X, \mathcal{D})$
4: $\{F, A\} :=$ `nondom`(Y, X)
5: $k := m$
6: **while** $k < n$ **do**
7: $\{\mathbf{T}, \boldsymbol{\tau}\} :=$ `sample`(A)
8: $\{\mathbf{T}', \boldsymbol{\tau}'\} :=$ `vary`$(\mathbf{T}, \boldsymbol{\tau}, \mathbf{x}^{max}, \mathbf{x}^{min})$
9: $\{\mathbf{y}'\} :=$ `evaluate`$(\{\mathbf{T}', \boldsymbol{\tau}'\}, \mathcal{D})$
10: $\{F, A\} :=$ `nondom`$(F \cup \{\mathbf{y}'\}, A \cup \{\mathbf{T}', \boldsymbol{\tau}'\})$
11: $k := k + 1$
12: **end while**
13: **return** $\{F, A\}$
14: **end procedure**

Algorithm 2 The variation mechanism.

1: **procedure** VARY($\mathbf{T}, \boldsymbol{\tau}, \mathbf{x}^{max}, \mathbf{x}^{min}$)
2: $\mathbf{T}' = \mathbf{T}$
3: $\boldsymbol{\tau}' = \boldsymbol{\tau}$
4: $r := \mathcal{U}(0, 1)$
5: **if** $r < 0.5$ **then**
6: $\boldsymbol{\tau}' :=$ τ`_perturb`$(\boldsymbol{\tau}', \mathbf{x}^{max}, \mathbf{x}^{min})$
7: $\mathbf{T}' :=$ `threshold_perturb`$(\mathbf{T}', \mathbf{x}^{max}, \mathbf{x}^{min})$
8: **else if** $r < 0.75$ **then**
9: $\boldsymbol{\tau}' :=$ τ`_perturb`$(\boldsymbol{\tau}', \mathbf{x}^{max}, \mathbf{x}^{min})$
10: **else**
11: $\mathbf{T}' :=$ `threshold_perturb`$(\mathbf{T}', \mathbf{x}^{max}, \mathbf{x}^{min})$
12: **end if**
13: **return** $\{\mathbf{T}, \boldsymbol{\tau}\}$
14: **end procedure**

Algorithm 3 Threshold variation.

1: **procedure** THRESHOLD_PERTURB($\mathbf{T}', \mathbf{x}^{max}, \mathbf{x}^{min}$)
2: $\{\mathbf{T}^{max}, \mathbf{T}^{min}\} :=$ `T_bounds`$(\mathbf{x}^{max}, \mathbf{x}^{min})$
3: $r := \mathcal{U}(0, 1)$
4: **if** $r < \frac{1}{3}$ **then**
5: $T_1' := \mathcal{G}(T_1', T_1^{max} - T_1^{min})$
6: $T_2' := \mathcal{G}(T_2', T_2^{max} - T_2^{min})$
7: **else if** $r < \frac{2}{3}$ **then**
8: $T_1' := \mathcal{G}(T_1', T_1^{max} - T_1^{min})$
9: **else**
10: $T_2' := \mathcal{G}(T_2', T_2^{max} - T_2^{min})$
11: **end if**
12: $\mathbf{T}' :=$ `truncate`$(\mathbf{T}', \mathbf{T}^{max}, \mathbf{T}^{min})$
13: **return** $\mathbf{T}'$
14: **end procedure**

Algorithm 4 Delay variation.

1: **procedure** τ_PERTURB($\boldsymbol{\tau}', \mathbf{x}^{max}, \mathbf{x}^{min}$)
2: $\{\boldsymbol{\tau}^{max}, \boldsymbol{\tau}^{min}\} :=$ τ`_bounds`$(\mathbf{x}^{max}, \mathbf{x}^{min})$
3: **if** $\mathcal{U}(0, 1) < 0.5$ **then**
4: $i := 1$
5: $j := 2$
6: **else**
7: $i := 2$
8: $j := 1$
9: **end if**
10: **if** $\mathcal{U}(0, 1) < 0.5$ **then**
11: $\tau_i' := \mathcal{G}(\tau_i', \tau_i^{max} - \tau_i^{min})$
12: $\tau_i' :=$ `truncate`$(\tau_i', \tau_i^{max}, \tau_i^{min})$
13: **end if**
14: **if** $\mathcal{U}(0, 1) < 0.5$ **then**
15: $\tau_j' := \mathcal{G}(\tau_j', \tau_j^{max} - \tau_j^{min})$
16: **end if**
17: $\tau_j' :=$ `truncate`$(\tau_j', \tau_j^{max} - \tau_i + 1, \tau_j^{min})$
18: **if** $\mathcal{U}(0, 1) < 0.5$ **then**
19: $\tau_3' := \mathcal{G}(\tau_3', \tau_3^{max} - \tau_3^{min})$
20: $\tau_3' :=$ `truncate`$(\tau_3', \tau_3^{max}, \tau_3^{min})$
21: **end if**
22: **return** τ'
23: **end procedure**