

Optimisation and Landscape Analysis of Computational Biology Models: A Case Study

Kevin Doherty
K.Doherty@exeter.ac.uk
University of Exeter, EX4 4QF, UK

Ozgur E. Akman*
O.E.Akman@exeter.ac.uk
University of Exeter, EX4 4QF, UK

Khulood Alyahya
K.Alyahya@exeter.ac.uk
University of Exeter, EX4 4QF, UK

Jonathan E. Fieldsend
J.E.Fieldsend@exeter.ac.uk
University of Exeter, EX4 4QF, UK

Abstract

The parameter explosion problem is a crucial bottleneck in modelling gene regulatory networks (GRNs), limiting the size of models that can be optimised to experimental data. By discretising state, but not time, Boolean delay equations (BDEs) provide a significant reduction in parameter numbers, whilst still providing dynamical complexity comparable to more biochemically detailed models, such as those based on differential equations. Here, we explore several approaches to optimising BDEs to timeseries data, using a simple circadian clock model as a case study. We compare the effectiveness of two optimisers on our problem: a genetic algorithm (GA) and an elite accumulative sampling (EAS) algorithm that provides robustness to data discretisation. Our results show that both methods are able to distinguish effectively between alternative architectures, yielding excellent fits to data. We also perform a landscape analysis, providing insights into the properties that determine optimiser performance (e.g. number of local optima and basin sizes). Our results provide a promising platform for the analysis of more complex GRNs, and suggest the possibility of leveraging cost landscapes to devise more efficient optimisation schemes.

CCS Concepts

•Applied computing → Biological networks; Systems biology; •Theory of computation → Evolutionary algorithms; •Computing methodologies → Uncertainty quantification;

Keywords

Systems biology, optimisation, landscape analysis, Boolean delay equations

ACM Reference format:

Kevin Doherty, Khulood Alyahya, Ozgur E. Akman, and Jonathan E. Fieldsend. 2017. Optimisation and Landscape Analysis of Computational Biology Models: A Case Study. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017*, 8 pages. DOI: <http://dx.doi.org/10.1145/3067695.3084609>

*Corresponding author

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

GECCO '17 Companion, Berlin, Germany

© 2017 Copyright held by the owner/author(s). 978-1-4503-4939-0/17/07...\$15.00
DOI: <http://dx.doi.org/10.1145/3067695.3084609>

1 Introduction

1.1 Optimising computational biology models

The development of computational models of gene regulatory networks (GRNs) is a crucial aspect of both systems and synthetic biology. Parameter optimisation is the key bottleneck in constructing such models [3, 11]. However, it is often addressed in an *ad hoc* manner, with parameters tuned by hand or optimised locally around previously published values, thereby placing strict limits on the size of systems that can currently be studied. Hence, there is a need for mathematical methods to reduce model complexity, and also for optimisation algorithms capable of dealing with highly parametrised problems of this type. In terms of the latter, standard approaches for optimising GRN models have not fully exploited the powerful techniques afforded by evolutionary computation, although promising results have been obtained by applying evolutionary algorithms (EAs) to a range of biological problems, such as the analysis and design of GRNs [12] and neural model fitting [5].

1.2 An exemplar GRN – the circadian clock

Circadian clocks are GRNs found in almost all organisms, controlling processes ranging from cyanobacterial cell division to human sleep-wake cycles [6]. These networks function by generating endogenous ~24 hour oscillations in gene expression (free-running rhythms) that can synchronise (entrain) to the external light-dark cycle. Entrainment enables organisms to time biochemical processes relative to dawn and dusk, providing an adaptive advantage [15]. The clocks of different organisms appear to have a similar structure based on interlocking gene-protein feedback loops [19].

Computational models of these feedback structures have proved useful in elucidating the general design principles of clocks, and also led to the discovery of novel circadian regulators [2, 11, 19]. The majority of clock models constructed thus far have been based on differential equations and possess on the order of 10 to 100 parameters, which is representative of the broader GRN problem class [3, 19]. A simple, representative model of this type is the minimal ordinary differential equation (ODE) model of the clock in the filamentous fungus *Neurospora crassa* [10], shown below:

$$\begin{aligned} \dot{M} &= (v_s + \theta(t)) \frac{k_I^N}{k_I^N + P_n^N} - v_m \frac{M}{k_m + M}, \\ \dot{P}_c &= k_s M - v_d \frac{P_c}{k_d + P_c} - k_1 P_c + k_2 P_n, \\ \dot{P}_n &= k_1 P_c - k_2 P_n. \end{aligned} \quad (1)$$

The model comprises three equations describing the dynamics of the oscillation-generating negative feedback loop: *FRQ* mRNA (M) is translated into protein (P_c) in the cytoplasm and then translocated into the nucleus (P_n), where it binds to the *FRQ* promoter and represses transcription. The model is parameterised by 10 kinetic constants: v_s – transcription rate; k_s – translation rate; v_m, v_d – degradation rates; k_1, k_2 – transport rates; k_I, k_m, k_d – activation/repression thresholds; N – promoter binding cooperativity. The function $\theta(t)$ models light, which upregulates *FRQ* transcription, thereby providing a mechanism for light entrainment [10]:

$$\theta(t) = \begin{cases} L_A & \text{if } t_{DAWN} < t \pmod{24} < t_{DUSK}; \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

In the above, L_A is the light amplitude (0 or 1) and t_{DAWN} and t_{DUSK} denote the times of dawn and dusk respectively.

1.3 GRN models based on Boolean delay equations

Modelling approaches based on Boolean logic provide a significant reduction in complexity compared to ODEs. 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 (logic gates) that calculate the state of a gene from the activation state of its upstream components [9, 17]. This approximation dramatically reduces both the number of system components and the number of system parameters [3].

In this study, we consider systems of Boolean delay equations (BDEs), which are parameterised by the collection of logic gates – referred to as the logic gate configuration – together with the set of signalling delays that specify the time it takes for state changes to take effect [4]. In [3], Akman *et al.* introduced a general scheme for modelling oscillators using the BDE formalism, constructing Boolean versions of several established ODE clock models. This included the *Neurospora* model (1), expressed in BDE form as

$$\begin{aligned} x_M(t) &= G(x_P(t - \tau_2), g_2) \text{ OR } \theta(t - \tau_3), \\ x_P(t) &= G(x_M(t - \tau_1), g_1), \end{aligned} \quad (3)$$

where x_M is *FRQ* mRNA, x_P is lumped *FRQ* protein (combining the cytoplasmic and nuclear forms), $\theta(t)$ is light and $\tau = (\tau_1, \tau_2, \tau_3)$ are the signalling delays. The function $G(x, g)$ implements the identity or NOT gate, modelling activation and repression by x respectively, depending on the value of the bit g : $G(x, 0) = x$, $G(x, 1) = \text{NOT } x$. The bitstring $\mathbf{g} = g_1g_2$ thus specifies the logic configuration, which is $\mathbf{g} = 01$ (activation of x_P by x_M ; repression of x_M by x_P). Note that there are 3 other possible configurations obtained by flipping g_1 and g_2 , which correspond to alternative regulatory structures consistent with the underlying directed graph of the ODE model (i.e. the graph with connections $M \rightarrow P_c \rightarrow P_n$ and $P_n \rightarrow M$): for example, $\mathbf{g} = 00$ corresponds to a circuit in which *frq* mRNA activates *FRQ* protein, but *FRQ* protein activates *FRQ* mRNA (a double positive feedback loop). Also note that system (3) has 2 variables and 3 parameters, compared with the 3 variables and 10 parameters of the ODE formulation. Note that to obtain a solution $\{\mathbf{x}(t) = (x_M, x_P(t)) : t \geq 0\}$ of (3) for a given \mathbf{g} and τ , it is necessary to specify an initial history, $\mathbf{x}_h = \{\mathbf{x}(t) : 0 \leq t \leq t_h\}$, where $t_h \geq \max(\tau)$ [3, 4].

1.4 Aims of the study

In [3], BDE versions of several established ODE clock models (including (1)) were optimised to timeseries data using a recursive grid search method. Here, we build on this previous work, leveraging evolutionary algorithms and landscape analysis to: (i) accelerate the optimisation process; and (ii) gain insight into how the regulatory structure of a model and the method used to score it against experimental data affect the underlying cost landscape.

2 Methods & Results

2.1 Data, discretisation and model prediction

Following the approach used in [3], we optimise the BDE model (3) to synthetic mRNA and protein data generated from the corresponding ODE formulation as follows: first, system (1) is integrated 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 are added together to obtain a bulk protein variable $P = P_c + P_n$; finally the mRNA and protein traces $\{(M(t), P(t)) : 0 \leq t \leq 120\}$ are sampled every 0.5h, mimicking the best possible experimental resolution [3].

In order to enable this continuous data to be compared to the solutions of (3) for a given logic configuration \mathbf{g} and delay set τ , the sampled timeseries are normalised between 0 and 1, and then discretised by applying thresholds $0 < T_1 < 1$ (mRNA) and $0 < T_2 < 1$ (protein), such that all subthreshold values are set to 0 (OFF) and all suprathreshold values to 1 (ON). E cycles of this discretised data, where $1 \leq E \leq 4$, are subsequently used to define the initial history for (3), by using linear interpolation to compute the time points within the history interval $0 \leq t \leq 24E$ at which switches in state occur. From this history, predicted mRNA and protein timeseries are generated over the time interval $24E \leq t \leq 120$ using a variant of the BDE solver described in [4]. The last S cycles of the model prediction, where $1 \leq S \leq 5 - E$, are then extracted, sampled every 0.5h, and scored against the discretised data lying in the same time interval ($120 - 24S \leq t \leq 120$), using one of the costing methods described in section 2.2. E cycles of data are thus used to generate the model prediction and S cycles are used to obtain a model score. We refer to this as an $E:S$ scoring protocol, and – following [3] – fix E to be 1, yielding 4 possible protocols: $E:S=1:4, 1:3, 1:2$ and $1:1$.

Replicating the approach used to fit ODE models [2, 11], we score the model against timeseries simulated in two light regimes: (i) constant darkness (termed DD), obtained by setting $L_A = 0$ in eqn. (2); and (ii) 12:12 light-dark (LD) cycles (alternating 12h periods of light and dark), obtained by setting $L_A = 1$, $t_{DAWN} = 6$ and $t_{DUSK} = 18$ in eqn. (2). The total score is then calculated as the sum of the DD and LD scores, as detailed in Section 2.2.

2.2 Optimisation experiments

The goal of our parameter optimisation is to determine the set of parameters giving the best fit of the model to the synthetic timeseries. We consider a number of ways of framing this problem: 1. For each fixed choice of gates $\mathbf{g} = g_1g_2$, we optimise over the combination of delays $\tau = (\tau_1, \tau_2, \tau_3)$ and thresholds $\mathbf{T} = (T_1, T_2)$; 2. We optimise over gates \mathbf{g} , delays τ and thresholds \mathbf{T} together; 3. We optimise over \mathbf{g} , τ and \mathbf{T} using fitness sharing between

solutions with the same \mathbf{g} to maintain diversity; 4. We repeat 1-3, but penalise threshold combinations yielding data with a low information content; 5. We repeat 1 & 2, calculating cost values for each combination of gates \mathbf{g} and delays $\boldsymbol{\tau}$ by averaging over thresholds. In cases 1-4, we optimise using a genetic algorithm (GA); in case 5, we use elite accumulative sampling (EAS) [7]. The two algorithms are described in detail in section 2.3.

2.2.1 Optimising with gates fixed In the first case, we solve the following constrained optimisation problem for each $\mathbf{g} \in \{0, 1\}^2$

$$\{\boldsymbol{\tau}, \mathbf{T}\}_{\min} = \arg \min_{\{\boldsymbol{\tau}, \mathbf{T}\}} C(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}), \quad 0 < \tau_k < 24, \quad \tau_1 + \tau_2 < 24, \quad (4)$$

$$C(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}) = \sum_{j=1}^2 \sum_{i=1}^2 d_H(\mathbf{F}_{ij}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{D}_j(\mathbf{T}, E), S), \mathbf{D}_{ij}(T_i, S)). \quad (5)$$

In the cost function (5) above, i indexes the model variables (FRQ mRNA or FRQ protein) and j indexes the light regimes (DD or LD). The constraints on the delays $\{\tau_k\}$ 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 [3]. For a given choice of j , $\mathbf{D}_j(\mathbf{T}, E)$ denotes the first E cycles of the timeseries obtained by discretising the synthetic data using thresholds $\mathbf{T} = (T_1, T_2)$; this is used as the initial history to generate a model prediction. The final S cycles of the predicted timeseries for each species i , denoted $\mathbf{F}_{ij}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{D}_j(\mathbf{T}, E), S)$, is then compared to the corresponding cycles of thresholded synthetic data, $\mathbf{D}_{ij}(T_i, S)$. The discrepancy d_H between prediction and data is measured as the Hamming distance between bitstrings \mathbf{F}_{ij} and \mathbf{D}_{ij} , normalised by bitstring length $|\mathbf{D}_{ij}|$. The overall cost C associated with the design $\{\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}\}$ is then obtained by summing over species i and light conditions j . Since $0 \leq d_H \leq 1$, it follows that $0 \leq C \leq 4$, with lower costs indicating better fits of the model to data.

2.2.2 Optimising over gates In the second case, we solve

$$\{\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}\}_{\min} = \arg \min_{\{\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}\}} C(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}), \quad 0 < \tau_k < 24, \quad \tau_1 + \tau_2 < 24. \quad (6)$$

2.2.3 Fitness sharing In the case of fitness sharing, we solve

$$\{\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}\}_{\min} = \arg \min_{\{\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}\}} C_{FS}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{T}), \quad 0 < \tau_k < 24, \quad \tau_1 + \tau_2 < 24. \quad (7)$$

Here, the cost of a particular solution C_{FS} is calculated as $4(1 - 1/n_g) + C/n_g$, where n_g is the number of solutions in the population with the same gate configuration as the solution in question. This penalises a solution if the population contains a relatively large number of solutions with the same gate configuration, thus encouraging the algorithm to maintain diversity in \mathbf{g} [16].

2.2.4 Penalising pathological solutions In solving problems (4), (6) and (7), it is possible to obtain designs with thresholds close to 0 or 1. In these cases, the discretised data contains bitstrings composed almost entirely of 0s or 1s. This yields solutions that, whilst having a low cost value, contain little temporal information (e.g. are non-oscillatory) [3]. To mitigate against this, we introduce a penalty term that assigns maximum cost to a variable if there is a significant probability of obtaining the same cost by chance.

The penalty term is calculated as follows. For each model variable i and light regime, j , let p_{ij} denote the proportion of ones in the data \mathbf{D}_{ij} and \hat{p}_{ij} the proportion of ones in the model prediction \mathbf{P}_{ij} . Then

the probability μ_{ij} of having a difference between the bitstrings at any location by chance is $\mu_{ij} = p_{ij} + \hat{p}_{ij} - 2p_{ij}\hat{p}_{ij}$. It follows that the total number of differences between the two bitstrings follows a Binomial distribution with mean $n_B\mu_{ij}$ and standard deviation $\sqrt{n_B\mu_{ij}(1 - \mu_{ij})}$, where $n_B = |\mathbf{D}_{ij}|$ is the bitstring length. Hence, for sufficiently large n_B , the normalised Hamming distance between the data and prediction which would be observed by chance is approximately normally distributed with mean μ_{ij} and standard deviation $\sigma_{ij} = \sqrt{\mu_{ij}(1 - \mu_{ij})/n_B}$. We use this approximation to calculate the probability of observing a Hamming distance equal to or lower than $d_H(\mathbf{F}_{ij}, \mathbf{D}_{ij})$ by chance. If this probability is less than 0.01, no penalty is applied; otherwise, maximum cost is assigned to variable i in light regime j by setting $d_H(\mathbf{F}_{ij}, \mathbf{D}_{ij}) = 1$ in (5).

2.2.5 Averaging over thresholds In generating a model score, the thresholds $\{T_1, T_2\}$ act as meta-parameters: although they don't directly affect the model (3) – which only depends on the delays $\boldsymbol{\tau}$ and logic gates \mathbf{g} – they do determine both the initial history used to compute the model prediction and the data which the prediction is costed against. In particular, extreme thresholds (i.e. T_i values near 0 or 1) can result in unphysical solutions. Consequently, it is desirable to find $\{\mathbf{g}, \boldsymbol{\tau}\}$ combinations which give predictions that are robust to uncertainty in the discretisation thresholds.

Here, we therefore also consider the following extension to problem (4): optimising over $\boldsymbol{\tau}$ for a fixed gate configuration \mathbf{g} , whilst calculating the average cost for individual solutions over a set of distinct threshold values. In this case, for each \mathbf{g} , we solve

$$\{\boldsymbol{\tau}\}_{\min} = \arg \min_{\boldsymbol{\tau}} C_A(\boldsymbol{\tau}, \mathbf{g}), \quad 0 < \tau_k < 24, \quad \tau_1 + \tau_2 < 24, \quad (8)$$

$$C_A(\boldsymbol{\tau}, \mathbf{g}) = \frac{1}{m} \sum_{j=1}^2 \sum_{l=1}^m \sum_{i=1}^2 d_H(\mathbf{F}_{ij}(\boldsymbol{\tau}, \mathbf{g}, \mathbf{D}_j(\mathbf{T}_l, E), S), \mathbf{D}_{ij}(T_{il}, S)). \quad (9)$$

In (9), the individual cost for each light regime ($j = 1, 2$) is calculated by averaging the sum of the variable scores over a fixed set of threshold combinations $\{\mathbf{T}_l = (T_{1l}, T_{2l}) : 1 \leq l \leq m\}$. We also consider the corresponding extension to problem (6):

$$\{\boldsymbol{\tau}, \mathbf{g}\}_{\min} = \arg \min_{\{\boldsymbol{\tau}, \mathbf{g}\}} C_A(\boldsymbol{\tau}, \mathbf{g}), \quad 0 < \tau_k < 24, \quad \tau_1 + \tau_2 < 24. \quad (10)$$

2.3 Optimisation algorithms

2.3.1 The genetic algorithm We use a genetic algorithm (GA) to solve: problem (4) – optimising over delays $\boldsymbol{\tau}$ and thresholds \mathbf{T} for fixed gates \mathbf{g} , referred to as experiment GA; problem (6) – optimising over $\boldsymbol{\tau}$, \mathbf{T} and \mathbf{g} together, referred to as experiment GA + G; and problem (7) – optimising over $\boldsymbol{\tau}$, \mathbf{T} and \mathbf{g} using fitness sharing, referred to as experiment GA + FS. In addition, we use the GA to solve the variants to these problems obtained by penalising thresholds yielding pathological solutions, referring to these as experiments GA + P, GA + G + P and GA + FS + P, respectively.

In each case, we use the function, `ga`, from the Global Optimization Toolbox in MATLAB 2016b with all options set to their default values, except for the following: (i) to initialise the population, we generate a uniform random sample of solutions, rather than implementing the default GA option `@gacreationlinearfeasible`, which creates a large number of individuals on the boundaries of the feasible region in parameter space; (ii) we have written a custom

mutation function in which a parameter is mutated with probability $1/d$, where d is the number of dimensions in parameter space, to ensure that on average one parameter is mutated. Real-valued parameters (delays and thresholds) are mutated by sampling from a normal distribution centred at the current value, with standard deviation set to 10% of the parameter range. Logical parameters (gates) are mutated by bit-flipping. Rejection sampling is performed to ensure new individuals satisfy bounds and constraints. We use a population size of 50 and run the algorithm for 49 generations, resulting in 2500 fitness evaluations. MATLAB's GA carries a number of elite solutions n_E from each generation to the next: we set n_E to 3 for fixed gate searches, and in the case of $GA + FS + P$, we keep 2 elite members for each gate configuration.

2.3.2 Elite accumulative sampling An elite accumulative sampling (EAS) algorithm was developed for problem (8) – optimising over delays τ for fixed gate configurations \mathbf{g} using threshold averaging, referred to as experiment *EAS*, and problem (10) – optimising over delays τ and configurations \mathbf{g} together using threshold averaging, referred to as experiment *EAS + G*.

The EAS algorithm addresses the arbitrariness in the choice of threshold parameters by searching for the best solution possible based on a running average cost computed over samples of threshold combinations (cf. cost function (9)). EAS records a history H of all points visited over the course of the algorithm. Individual solutions in H are sampled for at least one combination of thresholds and H is sorted according to solutions' average cost. Solutions are initially sampled at one threshold combination and solutions that have a low cost preferentially accumulate samples to progressively improve the estimate of their average cost. Not discarding any solutions means that we allow for a member of H with an initially poor fitness ranking to later improve its ranking as other solutions' average cost is increased upon resampling. The pseudocode for EAS is given in Algorithm 1.

We considered two approaches to sampling the thresholds T_i used in cost function (9): (i) each solution samples from the same sequence of thresholds; and (ii) each solution samples from independent threshold sequences. Independently sampling each solution resulted in a bias towards solutions that had been sampled in regions of T-space with good score (results not shown). We therefore decided to use a fixed set of thresholds, generated via Sobol sequences to ensure efficient space-filling [14]. This gave optimal solutions with uniformly distributed threshold sets, as desired.

2.4 Optimisation results

2.4.1 Optimising with fixed gates Table 1 summarises the results obtained from 30 runs each of experiments *GA*, *GA + P* and *EAS*. Of note is that when optimising using the *GA* with no penalty term included, the best solution found is for gate configuration $\mathbf{g} = 11$, rather than the configuration $\mathbf{g} = 01$ used to generate the data that is fitted to. This best solution has extreme thresholds, with $T_1 \approx 0$, $T_2 \approx 1$ (cf. Fig. 1), and is therefore unphysical. However, incorporating the penalty term (*GA + P*) prevents such pathological solutions being found, leading to the best solution being obtained for $\mathbf{g} = 01$, and also to a greater separation between the cost distributions by gate. Furthermore, the *EAS* results indicate that averaging over thresholds also yields a best solution with

Algorithm 1 Pseudo-code for the EAS algorithm. For the experiments presented here, we take $\text{max_evals} = 2500$, $\text{refine_evals} = 0.1\text{max_evals}$, $\text{num_elites} = 10$ and $\text{pop_size} = 50$.

```

Require: max_evals           Total number of evaluations.
Require: refine_evals       Number of evals. used for refinement.
Require: num_elites         Number of elite solutions.
Require: pop_size           Population size.
1:  $H \leftarrow \text{generate\_initial\_solutions}(\text{pop\_size})$    Initialise
   history with pop_size solutions.
2: num_evals = 0           Keep track of number of fitness evaluations.
3: for  $x \in H$  do
4:    $T_x = \text{sample\_thresh}(x)$ 
5:   evaluate( $x, T_x$ )
6:   num_evals = num_evals + 1
7: end for
8: while num_evals < max_evals do
9:   while num_evals < max_evals – refine_evals do
10:    pmut  $\leftarrow \text{draw\_random}(H[1 : \text{pop\_size}])$    Choose random
   parent from best pop_size solutions.
11:    cmut  $\leftarrow \text{mutate}(\text{pmut})$            Get child from mutation.
12:     $T_{\text{cmut}} \leftarrow \text{sample\_thresh}(\text{cmut})$ 
13:    evaluate( $\text{cmut}, T_{\text{cmut}}$ )
14:    num_evals  $\leftarrow \text{num\_evals} + 1$ 
15:    insert( $\text{cmut}, H$ )   Insert child into history and re-order.
16:    pxover1  $\leftarrow \text{draw\_random}(H[1 : \text{pop\_size}])$ 
17:    pxover2  $\leftarrow \text{draw\_random}(H[1 : \text{pop\_size}])$ 
18:    cxover  $\leftarrow \text{xover}(\text{p}_{\text{xover1}}, \text{p}_{\text{xover2}})$    Child from crossover
19:     $T_{\text{crossover}} \leftarrow \text{sample\_thresh}(\text{c}_{\text{xover}})$ 
20:    evaluate( $\text{c}_{\text{xover}}, T_{\text{crossover}}$ )
21:    num_evals  $\leftarrow \text{num\_evals} + 1$ 
22:    insert( $\text{c}_{\text{xover}}, H$ )   Insert child into history and re-order.
23:     $x_{\text{elite}} \leftarrow \text{remove\_least\_sampled}(H[1 : \text{num\_elites}])$    Get
   num_elites member with fewest threshold samples.
24:     $T_{x_{\text{elite}}} \leftarrow \text{sample\_thresh}(x_{\text{elite}})$ 
25:    evaluate( $x_{\text{elite}}, T_{x_{\text{elite}}}$ )
26:    num_evals  $\leftarrow \text{num\_evals} + 1$ 
27:    update_average_cost( $x$ )
28:    insert( $x, H$ )           Insert the resampled solution.
29:   end while
30:    $H_{\text{elite}} = \text{extract}(H[1 : \text{num\_elites}])$    Remove the num_elites
   best members from H.
31:    $x_{\text{elite}} \leftarrow \text{remove\_least\_sampled}(H_{\text{elite}})$    Get  $H_{\text{elite}}$ 
   member with fewest threshold samples.
32:    $T_{x_{\text{elite}}} \leftarrow \text{sample\_thresh}(x_{\text{elite}})$ 
33:   evaluate( $x_{\text{elite}}, T_{x_{\text{elite}}}$ )
34:   num_evals  $\leftarrow \text{num\_evals} + 1$ 
35:   update_average_cost( $x_{\text{elite}}$ )
36:   insert( $x, H_{\text{elite}}$ )   Insert the resampled solution, re-order.
37: end while
38: insert( $H_{\text{elite}}, H$ )           Insert the elite solutions.

```

$\mathbf{g} = 01$, and comparable separations between cost distributions by gate (albeit with greater spreads of scores and higher average costs).

The pronounced clustering of thresholds obtained in experiment *GA* for $\mathbf{g} = 01$ (cf. Fig. 1) lead us to investigate how the binary

Table 1: The mean cost, standard deviation (std) and best cost from 30 runs of GA, GA+P and EAS (optimisation with gates g fixed). For EAS, the results for each g are computed from the sampled threshold set yielding the lowest individual cost, associated with the best solution (lowest average cost).

		00	01	10	11
GA	mean	0.354	0.107	0.983	0.401
	std	0.190	0.039	0.124	0.172
	best	0.07	0.065	0.809	0.033
GA + P	mean	0.554	0.118	1.019	0.538
	std	0.055	0.058	0.228	0.024
	best	0.484	0.065	0.793	0.461
EAS	mean	0.653	0.343	1.017	0.637
	std	0.145	0.118	0.180	0.086
	best	0.308	0.166	0.867	0.422

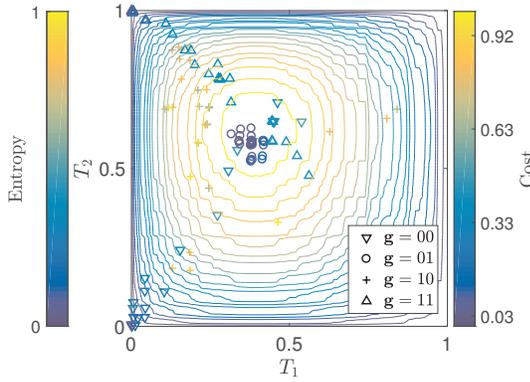


Figure 1: Threshold combinations for the optimal solutions found in 30 runs of experiment GA. Contours of the entropy function $\mathcal{E}(T)$ are shown for 20 equally spaced values between the minimum and maximum entropies. Thresholds are coloured by cost (right colourbar); contours by entropy (left colourbar). Lighter colours indicate higher values.

entropy of the data varies with the discretisation. We define the entropy for a given threshold combination $T = (T_1, T_2)$, $\mathcal{E}(T)$, as

$$\prod_{i=1}^2 \prod_{j=1}^2 \left(-p_{ij}(T_i) \log_2(p_{ij}(T_i)) - (1 - p_{ij}(T_i)) \log_2(1 - p_{ij}(T_i)) \right)$$

where $p_{ij}(T_i)$ is the proportion of ones in the data for variable i and light condition j when thresholded by T_i . It can be seen that for the data-generating gate configuration $g = 01$, the best scoring threshold combinations all have entropy values within 5% of the maximum value. Incorporating the penalty function further concentrates the thresholds around this maximum entropy point, and also pulls the best solutions for $g = 00$ and $g = 11$ away from extreme threshold combinations (results not shown).

2.4.2 Optimising with variable gates Figure 2 show the results obtained from 100 runs each of GA+G, GA+G+P, GA+FS+P and EAS + G. It can be seen that when optimising over g , τ and

T together using the GA, the majority (51%) of optimal solutions possess the data-generating gate configuration $g = 01$ (Fig. 2(d)), and that for each configuration returned – $g = 00, 01$ and 11 – the corresponding solutions lie on lines in τ -space of similar score (Figs. 2(a-b)). Also, whilst the best solutions with $g = 01$ are located in the high entropy regions of T -space, those with $g = 00$ and $g = 11$ possess extreme thresholds (Fig. 2(c)). Indeed, all solutions with $g = 01$ have a lower cost than any of those with $g = 00$ or $g = 11$, indicating that when searching over gates, the GA can get stuck in local minima associated with non data-generating configurations.

As expected from the fixed gate results, incorporating the penalty term into the GA cost function eliminates extreme thresholds, concentrating the distribution of optimal solutions in T -space around the maximum entropy point (Fig. 2(g)). Consequently, the GA becomes less likely to get stuck in local minima: configuration $g = 01$ is now optimal for 61% of runs (Fig. 2(h)) and the lines in τ space associated with each gate configuration are better separated and more homogenous ((Fig. 2(f))). Incorporating fitness sharing leads to even greater homogeneity in the optimal populations, with $g = 01$ now optimal for 96% of runs and $g = 00$ being eliminated from the set of optimal solutions altogether (Fig. 2(l)).

Interestingly, optimising over gates with EAS yields similar results to those obtained with the GA, although the threshold generates less well separated cost distributions for each gate configuration returned, with $g = 01$ only optimal for 31% of runs (Fig. 2(p)). However, despite this greater heterogeneity, $g = 01$ is still the gate associated with the best solutions overall, and also yields a lower average cost than the other configurations. Hence, EAS – like the GA – can also get stuck in local minima in this case.

2.5 Landscape analysis

The concept of a fitness landscape was first introduced by evolutionary biologist Sewall Wright [18] and has now spread to many other fields, including optimisation [13]. Formally, a landscape is a triple (X, N, f) , where X is the set of candidate solutions, N is a neighbourhood operator specifying the connectivity between candidate solutions and f is the objective function. Here, we were interested in studying the cost landscapes associated with our optimisation problem in order to quantify how changes in gate configuration g and scoring protocol $E:S$ affect the performance of our optimisers. To this end, we consider the landscapes obtained with X taken as τ -space and f defined as the weighted average over thresholds

$$f(\tau, g, E, S) = 4 - \int_0^1 \int_0^1 w(T_1, T_2) (4 - C(\tau, g, T_1, T_2)) dT_1 dT_2, \quad (11)$$

where C is as defined in eqn. (5) and $0 \leq w(T_1, T_2) \leq 1$ is the weighting function. Motivated by our optimisation results, we consider the landscapes arising when each point in τ -space is averaged over all thresholds equally ($w \equiv 1$), averaged with entropy-weighting ($w(T) = \mathcal{E}(T)$) and computed only at the thresholds (T_1^{\max}, T_2^{\max}) giving highest entropy ($w(T) = \delta(T_1 - T_1^{\max}, T_2 - T_2^{\max})$).

To calculate (11) for each choice of g , $E:S$ and w , we approximate the continuous delay space $\tau \in [0, 24]^3$ and threshold space $T \in [0, 1]^2$ as uniformly spaced grids with step sizes 0.5 and 0.05 respectively. We then compare the features of the landscapes in τ -space induced by the 1-neighbourhood and 2-neighbourhood operators, focussing on the following: the number of minima, the

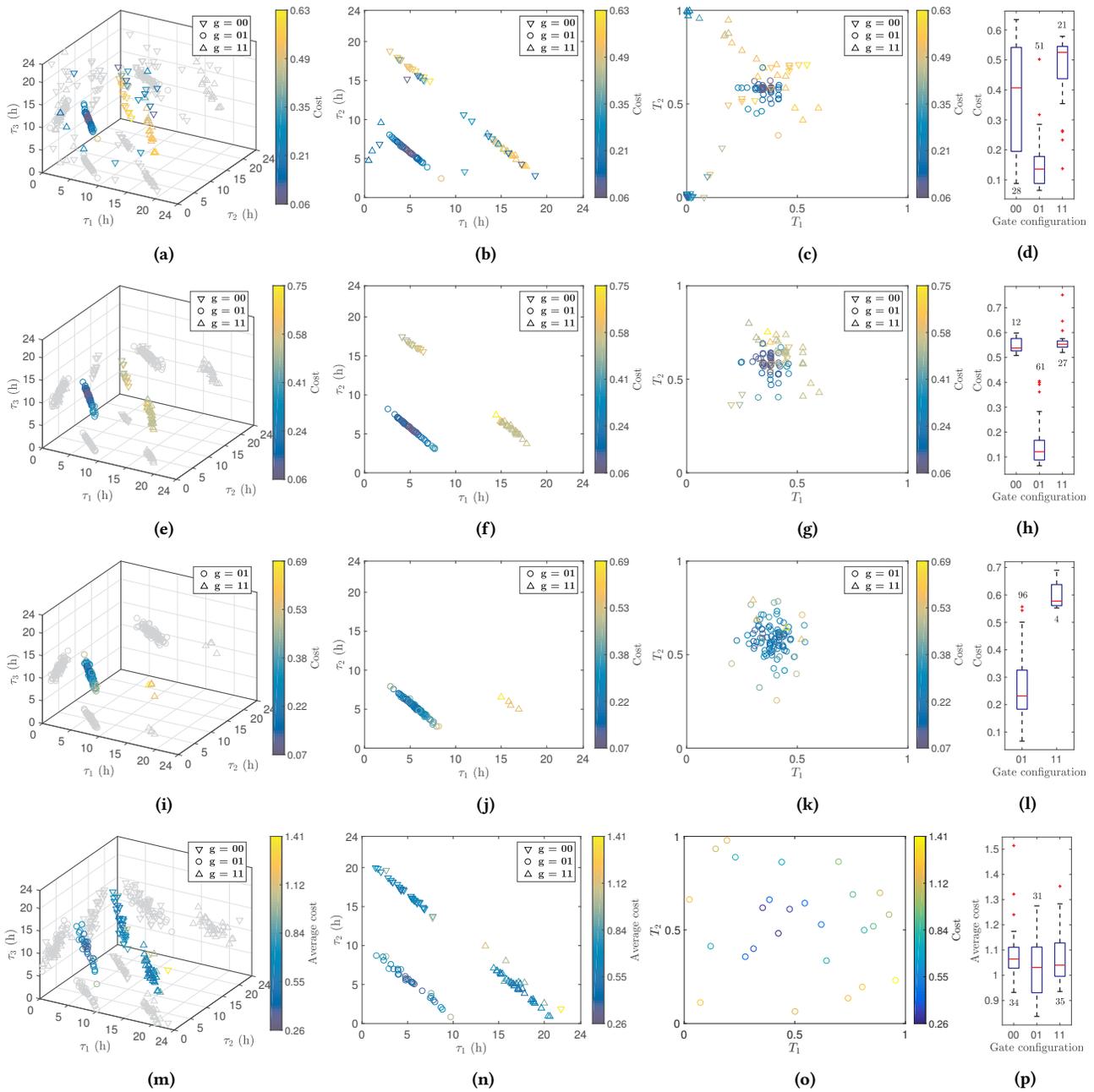


Figure 2: Best solutions obtained when the gate configuration is included in the optimisation. Results are shown for 100 runs each of GA+G (row #1), GA+G+P (row #2), GA+FS+P (row #3) and EAS+G (row #4), starting from the same initial population in each case. Columns #1-#3 show the projections of the solutions onto τ -space, the (τ_1, τ_2) -plane and T-space, respectively. Column #4 shows the distribution of cost values for each gate configuration returned (the number of solutions is indicated on the corresponding boxplot in each case). In (o), the T values plotted are those that the best solution was sampled over.

basin size, and the correlation between basin size and minimum cost. We define the basin of a local minimum as the number of feasible points $(\tau_1 + \tau_2 < 24)$ that lead to it under local search.

Fig. 3 shows that for a given gate combination g , the global minima obtained with the three different choices for threshold

weighting lie in close proximity to one another in τ -space (cf. the first two columns of Fig. 5). However, minima associated with different g values are separated in τ -space. We note that the minimum yielding the lowest overall cost is obtained for the data-generating gate $g = 01$ with the maximum entropy landscape. Assuming that

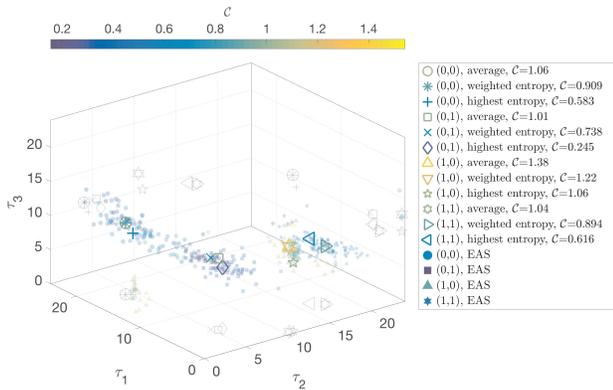


Figure 3: The global minima in the feasible region of τ -space ($\tau_1 + \tau_2 < 24$) obtained for each gate configuration when $E:S=1:4$, with the 3 different threshold averaging methods (equal weighting, entropy weighting, maximum entropy). Each minimum is coloured by cost (lighter colours denote higher cost values). The results of experiment *EAS* are also shown as filled symbols (cf. Table 1).

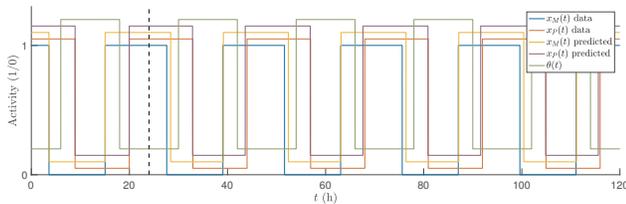


Figure 4: Sample timeseries generated by the model for optimal parameter sets. Discretised data and the corresponding model prediction are plotted in simulated LD for the best result found in experiment *GA + FS + P* ($g = 01$, $\tau_1 = 4.46$, $\tau_2 = 6.28$, $\tau_3 = 10.40$, $T_1 = 0.34$, $T_2 = 0.59$; $C_{FS} = 0.067$). The dashed line segregates history and scoring cycles (where $E:S = 1:4$); the LD cycle is also shown. Timeseries are offset for clarity.

EAS and *GA + P* are exploring the equal weighted and maximum entropy landscapes respectively, the apparent homology between the two landscapes may account for the similar results obtained for the two experiments (cf. Table 1). This assumption is supported by the proximity of the optimal solutions to the corresponding landscape minima (Fig. 3). Furthermore, the pronounced differences between the landscapes obtained for different g values may explain why, when searching across gates, the two algorithms are prone to getting stuck in local minima: mutations to g switch the landscape being searched to a significantly different one (e.g. compare columns #1 and #4 of Fig. 5). For the *GA*, fitness sharing evidently prevents this switching from compromising optimiser performance.

We note that although the weighting method has little impact on the positions of the global minima, the number of local optima found for maximum entropy landscapes is substantially higher than for the other two cases (Table 2). In addition, reducing the number of cycles S used for scoring results in the number of minima increasing

(e.g. compare columns #1 and #3 of Fig. 5, which shows that for $g = 01$, decreasing the number of cost cycles creates multiple valleys in the landscape). Finally, for all three weighting methods, we found the distribution of basin sizes to be very skewed, with most basins being of small size. In Table 2, we therefore also present the number of minima with large basins, determined by binning basin size using the Freedman-Diaconis rule [8] and then applying a 20% cut-off to the resulting distributions. In general, basin size and cost are correlated, with better minima having larger basins.

3 Discussion and Conclusions

In this work, we have presented new methods for optimising Boolean delay equation (BDE) systems modelling gene regulatory networks (GRN) to timeseries data. We discussed the advantages of BDEs compared to ordinary differential equations (ODEs); namely the dramatic reduction in parameters and the ability to systematically search over the finite set of alternative models (logic gate configurations) consistent with a given circuit diagram. We described how when fitting a particular gate configuration and delay set to data, it is necessary to specify discretisation thresholds – meta-parameters that also affect model score – thereby yielding an optimisation problem with 3 qualitatively different parameter sets.

To address these issues, we devised several optimisation schemes based on a standard genetic algorithm (GA) and a novel elite accumulative sampling (EAS) algorithm, where the latter addresses the arbitrariness in selecting discretisation thresholds by averaging over them. As a test case, we applied our methods to a BDE model of a simple circadian clock, fitting the model to synthetic data generated using the corresponding ODE model. We found that when optimising with gates fixed, the *GA* required a term penalising extreme thresholds for the best results to be obtained for the data-generating gate configuration $g = 01$; by contrast, the *EAS* algorithm consistently found best results for $g = 01$ (Table 1). When optimising over gates, both the *GA* and *EAS* algorithm were prone to getting stuck in local minima associated with alternative gate configurations. For the *GA*, fitness sharing mitigated against this, with $g = 01$ being returned as the optimal gate in almost all runs (Fig. 2). We note that for each optimisation scheme, the best individual solutions obtained had $g = 01$, and gave rise to near-perfect fits to the data (Fig. 4). These solutions lie very close to those obtained previously using a grid search method [3], but required significantly fewer function evaluations (2500 vs. 5.7×10^9).

In order to better understand the comparative performance of our optimisers, we performed a landscape analysis, computing cost surfaces in delay space by averaging over thresholds using different weighting functions. We found that different gate combinations give rise to landscapes possessing distinct structures (Fig. 5), hence providing insight into the inability of the *GA* (without fitness sharing) and *EAS* algorithm to consistently locate the global minimum when searching across gates. In addition, reducing the number of scoring cycles was found to significantly change the landscape properties, with an increase in the number of local optima and a weaker correlation between basin size and cost (Table 2). This finding is consistent with previous work showing that the performance of statistical parameter inference methods are highly sensitive to the number of circadian cycles used in the likelihood function [1].

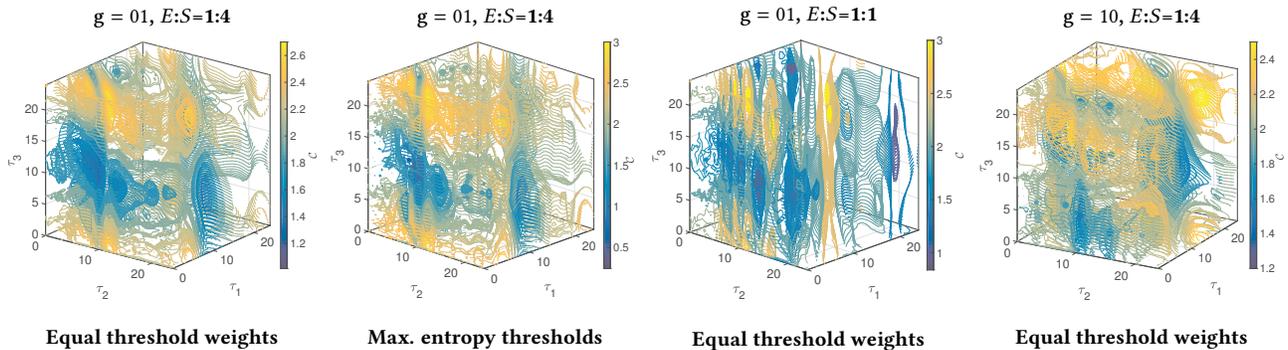


Figure 5: Sample cost landscape contour plots. Columns #1 and #2 show the landscapes obtained for $\{g = 01, E:S=1:4\}$ by averaging over thresholds with equal weighting and by selecting the thresholds with highest entropy, respectively. Columns #3 and #4 show the landscapes obtained with equal weight averaging for $\{g = 01, E:S=1:1\}$ and $\{g = 10, E:S=1:4\}$, respectively. The contours are shown for the entire search space, including the infeasible region $\tau_1 + \tau_2 > 24$. The minima located in the feasible region using the 1-neighbourhood operator are shown as spheres, with sizes scaled proportional to their basin size.

Table 2: Number of minima in the feasible region of τ -space ($\tau_1 + \tau_2 < 24$) induced by the 2-neighbourhood operator (values in brackets indicate the number of minima with basin sizes in the largest 80% of basin sizes). Results are shown for each gate configuration g , scoring protocol $E:S$ and threshold weighting method. A \dagger indicates that the best cost value lies in the infeasible region ($\tau_1 + \tau_2 > 24$). A \ddagger indicates a weak correlation between basin size and cost (Kendall's $\tau_k < -0.3$).

	Equal threshold weights				Entropy weighted thresholds				Maximum entropy thresholds			
	1:4	1:3	1:2	1:1	1:4	1:3	1:2	1:1	1:4	1:3	1:2	1:1
00	67 (2)	206 (2) \ddagger	300 (1)	308 (3) \ddagger	86 (1)	180 (1) \ddagger	281 (2)	350 (4) \ddagger	6785 (7)	7198 (2) \ddagger	7408 (3)	7938 (2) \ddagger
01	56 (4)	48 (6)	66 (5)	41 (6) \ddagger	49 (5)	52 (5)	67 (6)	49 (7) \ddagger	103 (13)	143 (5) \ddagger	198 (11)	183 (10) \ddagger
10	62 (6) \ddagger	57 (4) \ddagger	78 (4) \ddagger	51 (6) \ddagger	54 (8) \ddagger	53 (6) \ddagger	71 (4) \ddagger	53 (12) \ddagger	135 (7) \ddagger	157 (6) \ddagger	234 (7) \ddagger	237 (14) \ddagger
11	54 (2)	271 (2) \ddagger	314 (2) \ddagger	369 (5) \ddagger	71 (1)	263 (2) \ddagger	428 (2)	417 (5) \ddagger	994 (4)	875 (1) \ddagger	1515 (4)	2030 (5) \ddagger

In summary, this study provides a quantitative framework for optimising BDE models of GRNs. Future work includes: (i) scaling our methods to larger, more complex models, and compensating for the poor scaling of grids with problem size; (ii) incorporating fitness sharing into the EAS algorithm; and (iii) applying landscape analysis to experimentally recorded circadian rhythms to quantify how data uncertainty impacts optimiser performance.

Acknowledgments

This work was financially supported by the Engineering and Physical Sciences Research Council [grant numbers EP/N017846/1, EP/N014391/1], and made use of the Zeus and Isca supercomputing facilities provided by the University of Exeter HPC Strategy.

References

- [1] S. Aitken and O. E. Akman. 2013. Nested sampling for parameter inference in systems biology: application to an exemplar circadian model. *BMC Syst. Biol.* 7, (2013), 72.
- [2] O. E. Akman, D. A. Rand, P. E. Brown, and A. J. Millar. 2010. Robustness from flexibility in the fungal circadian clock. *BMC Syst. Biol.* 4 (2010), 88.
- [3] O. E. Akman et al. 2012. Digital clocks: simple Boolean models can quantitatively describe circadian systems. *J. Roy. Soc. Interface* 9, 74 (2012), 2365–2382.
- [4] D. P. Dee and M. Ghil. 1984. Boolean Difference Equations, I: Formulation and Dynamic Behavior. *SIAM J. Appl. Math.* 44, 1 (1984), 111–126.
- [5] S. Druckmann et al. 2007. A novel multiple objective optimization framework for constraining conductance-based neuron models by experimental data. *Front. Neurosci.* 1, 1 (2007), 7–18.
- [6] J. C. Dunlap, J. J. Loros, and P. J. DeCoursey. 2003. *Chronobiology: Biological Timekeeping*. Sinauer.
- [7] J. E. Fieldsend and R. M. Everson. 2015. The Rolling Tide Evolutionary Algorithm: A Multiobjective Optimizer for Noisy Optimization Problems. *IEEE Trans. Evol. Comp.* 19, 1 (2015), 103–117.
- [8] D. Freedman and P. Diaconis. 1981. On the histogram as a density estimator: L_2 theory. *Z. Wahrsch. Verw. Gebiete* 57, 4 (1981), 453–476.
- [9] S. A. Kauffman. 1969. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* 22, 3 (1969), 437–467.
- [10] J. C. Leloup, D. Gonze, and A. Goldbeter. 1999. Limit Cycle Models for Circadian Rhythms based on Transcriptional Regulation in *Drosophila* and *Neurospora*. *J. Biol. Rhythms* 14, 6 (1999), 433–448.
- [11] J. C. W. Locke et al 2006. Experimental validation of a predicted feedback loop in the multi-oscillator clock of *Arabidopsis thaliana*. *Mol. Syst. Biol.* 2 (2006), 59.
- [12] I. Otero-Muras and J. R. Banga. 2014. Multicriteria global optimization for biocircuit design. *BMC Syst. Biol.* 8 (2014), 113.
- [13] E. Pitzer and M. Affenzeller. 2012. A Comprehensive Survey on Fitness Landscape Analysis. In *Recent Advances in Intelligent Engineering Systems*, J. Fodor, R. Klemous, S. Araujo, and C. Paz (Eds.). Vol. 378. Springer, Berlin, 161–191.
- [14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing* (2nd ed.). CUP, Cambridge.
- [15] V. Resco, J. Hartwell, and A. Hall. 2009. Ecological implications of plants ability to tell the time. *Ecol. Lett.* 12, 6 (2009), 583–592.
- [16] B. Sareni and L. Krahenbuhl. 1998. Fitness Sharing and Niching Methods Revisited. *IEEE Trans. Evol. Comput.* 2, 3 (1998), 97–106.
- [17] R. Thomas. 1991. Regulatory networks seen as asynchronous automata: A logical description. *J. Theor. Biol.* 153 (1991), 1–23.
- [18] S. Wright. 1932. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proc. 6th Int. Congr. on Genetics*, Vol. 1. 356–366.
- [19] E. E. Zhang and S. A. Kay. 2010. Clocks not winding down: unravelling circadian networks. *Nat. Rev. Mol. Cell Biol.* 11, 11 (2010), 764–776.