Computational design of synchronous sequential structures in biological systems

Supplementary text

Lidija Magdevska, Žiga Pušnik, Miha Mraz, Nikolaj Zimic, Miha Moškon*

[1] Faculty of Computer and Information Science, University of Ljubljana, Večna pot 113, SI-1000 Ljubljana, Slovenia

* corresponding author
e-mail: miha.moskon@fri.uni-lj.si

1 Modelling the dynamics of gene regulatory networks

Quantitative modelling of gene regulatory networks (GRNs) is usually performed using a system of coupled ordinary differential equations (ODEs) [1, 2]. These approximate an average response of the system in a deterministic manner [3]. Each equation in the system describes the dynamics of a specific chemical species in the following way:

$$\frac{dx_i}{dt} = f_i(\mathbf{x}(t), \mathbf{p}), \tag{1}$$

where f_i is a function governing the dynamics of chemical species x_i ($i \in \{1, 2, ..., n\}$) in dependence on the state of the system in time step t, i.e. $\mathbf{x}(t) = (x_1(t), x_2(t), ..., x_n(t))$ and in dependence on model parameters $\mathbf{p} = (p_1, p_2, ...)$. In the most simple scenario these functions usually approximate two cellular processes, i.e. *gene expression* and *protein degradation.* While protein degradation can be described with the simple mass action kinetics, where degradation is proportional to protein concentration and its degradation rate, Hill equations are used to describe the regulated gene expression [4]. Let's presume that the expression of protein x is regulated with transcription factor (TF) y. We can describe the dynamics of x with the following equation

$$\frac{dx}{dt} = \begin{cases} \alpha \cdot \frac{\left(\frac{y}{K_d}\right)^n}{1+\left(\frac{y}{K_d}\right)^n} - \delta \cdot x, \text{ when } y \text{ is activator,} \\ \alpha \cdot \frac{1}{1+\left(\frac{y}{K_d}\right)^n} - \delta \cdot x, \text{ when } y \text{ is repressor,} \end{cases} \tag{2}$$

where x and y describe the protein concentrations, α maximal protein expression rate, K_d dissociation constant between the TF y and promoter binding sites, n

Hill coefficient and δ degradation rate of protein x. Gene expression dynamics can be in several cases approximated with a *unit step* or *Heaviside function*, which means that the gene is either *fully off* or *maximally on* [4]. Applying the approximation to the Equation 2 yields the following equation

$$\frac{dx}{dt} = \begin{cases} \alpha \cdot \Theta(y - K_d) - \delta \cdot x, \text{ when } y \text{ is activator,} \\ \alpha \cdot \Theta(K_d - y) - \delta \cdot x, \text{ when } y \text{ is repressor,} \end{cases} \tag{3}$$

where Θ is a unit step function returning 0 when its argument is a negative value, and 1 when its argument is a positive value. This notation can be extended to multiple TFs regulating the expression of the same gene in the following way

$$\begin{aligned} \frac{dx}{dt} = \quad & \alpha \cdot (\Theta(a_1 - K_{d_{a1}}) \cdot \Theta(a_2 - K_{d_{a2}}) \cdot ... \cdot \Theta(a_k - K_{d_{ak}}) \cdot \\ & \Theta(r_1 - K_{d_{r1}}) \cdot \Theta(r_2 - K_{d_{r2}}) \cdot ... \cdot \Theta(a_k - K_{d_{rl}})) - \delta \cdot x, \end{aligned} \tag{4}$$

where a_1, a_2,..., a_k are transcriptional activators and r_1, r_2,..., r_l are transcriptional inhibitors.

2 Kinetic parameters values

The kinetic parameter ranges used in the process of tuning were derived from the basic quantitative properties of the bacterial *E. coli* cell [4] (see Table 1). In order to do so we converted the quantitative properties of the cell to kinetic

property	value
time to transcribe a gene (T_{trsc})	$\sim 1\ min = 60\ s$
time to translate a protein (T_{trsl})	$\sim 2\ min = 120\ s$
mRNA half-life ($T_{1/2_{mRNA}}$)	$\sim 3.5\ min = 210\ s$
cell half-life ($T_{1/2_{cell}}$)	$\sim 30\ min = 1800\ s$

Table 1: Basic quantitative properties of the bacterial *E. coli* cell according to [4].

rates that are used in our models. The calculation of the transcription (k_{trsc}) and translation rates (k_{trsl}) can be calculated with the following equations:

$$k_{trsc} = \frac{1}{T_{trsc}}, \tag{5}$$

$$k_{trsl} = \frac{1}{T_{trsl}}. \tag{6}$$

The calculation of the mRNA degradation rate is performed with the following equation:

$$\delta_{mRNA} = \frac{\ln(2)}{T_{1/2_{mRNA}}}. \tag{7}$$

We presume that the average protein half-lives equal the average cell half-life. The protein degradation rate can be thus expressed as:

$$\delta = \frac{\ln(2)}{T_{1/2_{cell}}}. \tag{8}$$

Since we model the transcription and translation as a single step reaction we need to calculate its kinetic rate, which can be expressed as:

$$\alpha = \frac{k_{trsc} \cdot k_{trsl}}{\delta_{mRNA}}. \tag{9}$$

The reference parameter values obtained from the basic quantitative properties of *E. coli* described in Table 1 and Equations 5–9 are described in Table 2. We

parameter	description	reference value
k_{trsc}	transcription rate	$1.6667 \cdot 10^{-2}\ s^{-1}$
k_{trsl}	translation rate	$8.3333 \cdot 10^{-3}\ s^{-1}$
δ_{mRNA}	mRNA degradation rate	$3.3007 \cdot 10^{-3}\ s^{-1}$
δ	protein degradation rate	$3.8508 \cdot 10^{-4}\ s^{-1}$
K_d	dissociation constant	$1\ nM$
α	gene expression rate	$4.2079 \cdot 10^{-2}\ s^{-1}$

Table 2: Reference parameters values.

still need to define the ranges of actual kinetic parameter values that are used in the process of the model response optimisation. For these, we presume the following ranges:

- δ: from 10^{-1}–fold to 10^{1}–fold its reference value;
- K_d: from 10^{-2}–fold to 10^{2}–fold its reference value;
- α: from 10^{-1}–fold to 10^{1}–fold its reference value.

3 Genetic algorithm

Genetic algorithm (GA) was used to tune the dynamic response of proposed master-slave flip-flop implementation [5]. Each optimisation was performed with 16 iterations of the GA using 40 individuals. Initial parameter values were set to random values from the intervals described in Section 2. Further optimisation was performed with the evaluation of *fitness function* of each individual solution, *mutation* of individual solutions and their *selection*.

3.1 Cost function

We assumed that high protein concentration should equal 100 nM and low protein concentration maximally 10 nM. We defined a sequence of desired states for each of the described topologies. In the first topology, where q is the input data protein, we expected a high protein q concentration during all four clock periods. In the second topology, where q_c is the input data protein, we expected the q concentration to be high during the first clock period, low during the second, high during the third and low during the fourth. The cost of each individual solution was determined as the sum of partial costs that were calculated for each of the four periods separately. The partial costs were calculated in k time points for both output proteins, i.e. q and q_c.

The optimisation was divided in two phases. In the first 80 % of optimisation time mean values of proteins q and q_c were compared to their expected concentrations in k time points. In the remaining 20 % of optimisation time variance of proteins q and q_c was also observed to fine tune the dynamic response of the system. If the calculated partial costs of an individual exceeded a predefined threshold (i.e. 10 nM), it was artificially increased to lower the probability for the individual to be selected into the next round of optimisation.

3.2 Mutations

We defined an interval $[x, y]$, which was linearly narrowed after each iteration. When performing mutations on an individual, we randomly select one of its parameters, multiply it by a random value from the interval $[0, y - x]$, and finally add value x. If the upper or lower parameter boundary is exceeded, we set the parameter to the exceeded boundary value. For the starting interval $[x, y]$ interval $[0.05, 1.95]$ was selected, while for the ending interval $[0.4, 1.6]$ was used.

3.3 Selection

Only half of the population will survive and thus remain in the next round of the optimisation. All individuals from the current population are sorted in an increasing order regarding their costs. The fittest one percent (rounded up) will always survive (*elitism*). The remaining individuals are selected with a roulette rule. The probability that the individual i is selected equals:

$$p_i = \frac{i}{\frac{n \cdot (n+1)}{2}}, \tag{10}$$

where i is the index of an individual in the sequence of sorted individuals, and n is the population size. Each of the individuals can be selected more than once. Individuals are selected according to their probabilities as long as necessary, i.e. until all positions (i.e. half of the new population) are taken. The remaining half of the population in the new generation will be generated with the mutations (see Section 3.2) of the first half of the population.

4 Description of Matlab code

The code used in this paper is available at http://lrss.fri.uni-lj.si/bio/material/counter.zip under the Creative Commons Attribution license. Below is its brief documentation.

- 'simulations\find_params_d.m': finds a set of kinetic parameter values for which the flip-flop topology exhibits predefined dynamics.
- 'simulations\counter.m': performs a simulation of Johnson counter with the results of 'find_params_d.m'.
- 'simulations\getParametersRange.m': returns valid ranges for each kinetic parameter.
- 'simulations\plotGraphs.m: plots and saves flip-flop simulation graphs obtained with the results of 'find_params_d.m'.
- 'analysis\runSimulations.cmd': Windows batch script that runs 20 flip-flop simulations in parallel and saves the results to a given subfolder.
- 'analysis\find_params.m': finds a set of kinetic parameter values for which the flip-flop topology exhibits predefined dynamics and saves the results to a given subfolder.
- 'analysis\morris.m': performs the Morris sensitivity analysis.
- 'analysis\drawErrorBarSensitivity.m': draws the error bars for sensitivity analysis results.
- 'analysis\drawHeatMaps.m': performs the parameter sweep analysis and saves generated heatmaps to subfolder 'heat'.

References

[1] N. Le Novere, "Quantitative and logic modelling of molecular and gene networks," *Nature Reviews Genetics*, vol. 16, no. 3, pp. 146–158, 2015.

[2] H. de Jong, "Modeling and simulation of genetic regulatory systems: a literature review.," *Journal of Computational Biology*, vol. 9, no. 1, pp. 67–103, 2002.

[3] M. Kaern, W. J. Blake, and J. Collins, "The engineering of gene regulatory networks," *Annual Review of Biomedical Engineering*, vol. 5, pp. 179–206, 2003.

[4] U. Alon, *An Introduction to Systems Biology.* Chapman & Hall/CRC, 2007.

[5] C. H. J. Sun, J.M. Garibaldi, "Parameter estimation using metaheuristics in systems biology: A comprehensive review," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, pp. 185–202, 2012.