Publication: Principles of Computation by Competitive Protein Dimerization Networks

Authors: Jacob Parres-Gold, Matthew Levine, Ben Emert, Andrew Stuart, Michael B. Elowitz

Primary contact for code & data: Jacob Parres-Gold, jacobparresgold@gmail.com

This dataset is intended to demonstrate how the data were generated and how to re-analyze the data if necessary. All analysis was originally performed on an Amazon Web Services (AWS) c5.4xlarge instance and has been further annotated for readability. We tried to carefully implement as few changes as possible; however, it is always possible that bugs may have been introduced, such as in file-handing commands. See the Methods section for details on the analysis.

Utilities

The following Python script provides functions that are used by other notebooks in the dataset:

- `dimer_network_utilities.py`

Resources

The following resources are available to simulate the input-output functions of arbitrary networks:

`simulate_individual_networks.ipynb`	This notebook demonstrates simulations of individual one- and two-input networks with schematics of the network architecture.
Interactive Google Colab	Interactive Google Colab notebook for simulating one- and two-input networks.

Notebooks to Replot Figures

The following notebooks were used to plot the figures used in the paper. They were created to make re-plotting of the data as simple as possible. The data for these figures are drawn directly from the archived data folder. These notebooks also include many plots not included in the main or supplementary figures; such plots have not been aesthetically cleaned up.

`remake_GraphicalAbstract.ipynb`	Plots for graphical abstract
`remake_Fig1.ipynb`	Plots for Figure 1
`remake_Fig2.ipynb`	Plots for Figure 2
`remake_Fig3.ipynb`	Plots for Figure 3
`remake_Fig4.ipynb`	Plots for Figure 4 and Figure S3
`remake_Fig5.ipynb`	Plots for Figure 5 and Figure S4
`remake_Fig6.ipynb`	Plots for Figure 6 and Figure S6
`remake_FigS5.ipynb`	Plots for Figure S5
`remake_FigS7.ipynb`	Plots for Figure S7

Notebooks to Demonstrate Analysis Methods

The following notebooks were created from code that was used to perform the majority of the analysis for the paper. Many of these notebooks use the ray package for parallelization, which is not available for Windows. Thus, if you would like to re-run these notebooks, make sure to disable the commands involving ray. It may also be necessary to change code relevant to loading and saving files.

`generate_param_screen.ipynb`	This notebook performs a parameter screen of networks, considering 1 or 2 monomers as the input.
`analyze_1D_screen.ipynb`	This notebook analyzes the responses observed in the parameter screen of one-input network responses.
`analyze_2D_screen.ipynb`	This notebook analyzes the responses observed in the parameter screen of two-input network responses.
`analyze_connectivity_trends.ipynb`	Using the large screen of many networks, this notebook assesses how various network properties vary with network connectivity.
`optimizer_dualannealing.ipynb`	This notebook demonstrates how a dual annealing algorithm was used to optimize dimerization networks to perform desired functions.
`boolean_complexity.ipynb`	This notebook demonstrates how to calculate the Boolean complexity of the Boolean functions used in this work.
`analyze_equilibration_kinetics.ipynb`	This notebook uses deterministic simulations, via numerical integration of ordinary differential equations (ODEs), to estimate the time required for network re-equilibration after a perturbation of the input monomer.
`analyze_intrinsic_noise.ipynb`	This notebook will use stochastic stimulations, via Gillespie simulation, of networks at steady state to estimate stochastic fluctuations in dimer concentrations at equilibrium.
`analyze_expression_noise_robustness.ipynb`	This notebook simulates various networks (one for each unique function) under random perturbations of monomer expression levels (total abundances).

Data: General

The full dataset is quite large (~332 Gb) and has been combined into a .tar file, which can be opened using the following instructions. Search online in case any syntax has changed since the release of this dataset.

- In MacOS, you may be able to open the tar file in Finder.
- Otherwise, you can use the following command in Command Prompt (Windows) or Terminal (Mac) after navigating to the proper folder (directory).
    - `tar -xvf parresgold_2023_dimer_networks_data.tar`

We have also created a more user-friendly subset of the data in a zip file called `parresgold_2023_dimer_networks_plotting_data.zip`. This file contains only the data necessary to re-plot the figures and is significantly smaller in size.

Data from Parameter Screens

`param_screen_1D`	Raw data from one-input parameter screen.
`param_screen_1D_limited_param_range`	Raw data from one-input parameter screen with a more limited parameter range.
`param_screen_2D`	Raw data from two-input parameter screen.
`param_screen_analysis_1D`	Results of downstream analysis of the one-input parameter screens.
`param_screen_analysis_1D_limited_param_range`	Results of downstream analysis of the one-input parameter screens with a more limited parameter range.
`param_screen_analysis_2D`	Results of downstream analysis of the two-input parameter screens.

Data from Annealing Optimizations for Multi-Input Logic Gates

`optimization_trials_high_dim_logic_gates`	Optimization results, based on the dual annealing optimizer, for three- and four-input optimization trials.

Data on Transcription Factor Co-Expression

`transcription_factor_coexpression`	Co-expression data for nuclear receptor (NR) and bZIP transcription factors in mouse and humans.

Code and Data for Versatility Analysis

The code for the versatility analysis, in which genetic algorithm optimization was used to optimize accessory expression levels of networks to perform desired functions, was primarily written by

Matthew Levine. This code was not condensed like the above notebooks were; instead, they have been directly included as-is. They can be found in the folders listed below, as well as on GitHub. Information about each file can be found on the GitHub. This code, rather than being run in a notebook, was run as a series of batch scripts. We understand that this code is not as easily parsed as the rest; if you have questions, please reach out to the authors.

`optimization_trials_1D`	Versatility results for the general one-input functions.
`optimization_trials_1D_testing_connectivity`	Versatility results for an experiment in which we tested networks of 8 monomers while explicitly varying network connectivity.