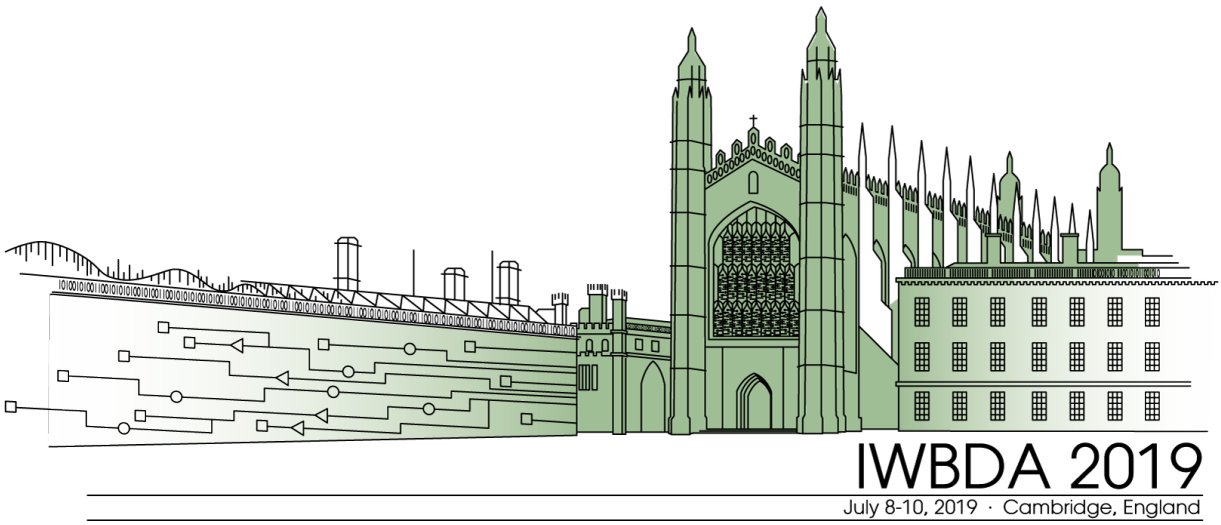


11th International Workshop on Bio-Design Automation
University of Cambridge, UK
July 8th-10th 2019

Foreword

Welcome to IWBDA 2019!

The IWBDA 2019 Executive Committee welcomes you to Cambridge, UK, for the Eleventh International Workshop on Bio-Design Automation (IWBDA). IWBDA brings together researchers from the synthetic biology, systems biology, and design automation communities. The focus is on concepts, methodologies, and software tools for the computational analysis and synthesis of biological systems.

The field of synthetic biology, still in its early stages, has largely been driven by experimental expertise, and much of its success can be attributed to the skill of the researchers in specific domains of biology. There has been a concerted effort to assemble repositories of standardized components; however, creating and integrating synthetic components remains an ad hoc process. Inspired by these challenges, the field has seen a proliferation of efforts to create computer-aided design tools addressing synthetic biology's specific design needs, many drawing on prior expertise from the electronic design automation (EDA) community. IWBDA offers a forum for cross-disciplinary discussion, with the aim of seeding and fostering collaboration between the biological and the design automation research communities.

IWBDA is proudly organized by the non-profit Bio-Design Automation Consortium (BDAC). BDAC is an officially recognized 501(c)(3) tax-exempt organization.

This year, the program consists of 19 contributed talks and 12 poster presentations. The talks are organized into 6 sessions: Design Automation, Machine-Learning, Computation in Genetic Circuits, Modelling, Data standards and visualization, and Microfluidics. In addition, we are very pleased to have two distinguished invited speakers: Prof. Luca Cardelli from Univeristy of Oxford and Dr. Traci Haddock-Angelli from the iGEM Foundation.

We would like to thank all the participants for contributing to IWBDA. We would also like to thank the Program Committee for reviewing the abstracts and everyone on the Executive Committee for their time and dedication. Finally, we would like to thank NSF, BBN Technologies, Lattice Automation, Agilent, and ACS Synthetic Biology for their support.

Sponsors

“Design” Level

“Algorithm” Level

Raytheon
BBN Technologies

The following participants were provided financial support by our sponsors to attend IWBDA 2019

Radhakrishna Sanka	Boston University
David McIntyre	Boston University
Ayush Pandey	California Institute of Technology
Jeanet Mante	University of Utah
Pedro Fontanarrosa	University of Utah
Sanaullah	Chosun University
Guillermo Yanez	Pontificia Universidad Catolica de Chile
Javier Carrión	Universidad Diego Portales
Yerko Ortiz	Universidad Diego Portales
Iuliia Zarubiieva	University of Warwick
Xinwu Yu	University of Warwick
Francesca Mantellino	University of Warwick
Uriel Urquiza-García	The University of Edinburgh
Matthew Jones	Cambridge Consultants

Organizing Committee

Executive Committee

Co-General Chair - Pietro Lio', University of Cambridge

Co-General Chair - Anil Wipat, Newcastle University

Co-General Chair - Jim Haseloff, University of Cambridge

Co-General Chair - Andrew Phillips, Microsoft Research

Co-General Chair - Sara-Jane Dunn, Microsoft Research

Co-Local Chair - Alexandra Ting, University of Cambridge

Co-Local Chair - Helena Andres Terre, University of Cambridge

Co-Local Chair - Zuliani Paolo, Newcastle Cambridge

Program Committee Chair - Prashant Vaidyanathan, Microsoft Research

Publication Chair - Prashant Vaidyanathan, Microsoft Research

Co-Web Chair - Aaron Adler, BBN Technologies

Co-Web Chair - Prashant Vaidyanathan, Microsoft Research

Finance Chair - Traci Haddock-Angelli, iGEM Foundation

Bio-Design Automation Consortium

President - Douglas Densmore, Boston University

Vice-President - Aaron Adler, BBN Technologies

Treasurer - Traci Haddock, iGEM Foundation

Clerk - Natasa Miskov-Zivanov, Carnegie Mellon University, Clerk

Program Committee

Ernst Oberortner	DOE Joint Genome Institute
Sara-Jane Dunn	Microsoft
Prashant Vaidyanathan	Microsoft
Xianwei Meng	DOE Joint Genomic Institute
Bradley Brown	Newcastle University
Luis Ortiz	Boston University
Chris Myers	University of Utah
Paolo Zuliani	Newcastle University
Anil Wipat	Newcastle University
Jacob Beal	BBN Technologies
Evan Appleton	Havard Medical School
Göksel Mısırlı	Keele University
Andrew Phillips	Microsoft
Aaron Adler	BBN Technologies
Nicholas Roehner	Boston University
James McLaughlin	Newcastle University
Eric Young	Worcester Polytechnic Institute
Natasa Miskov-Zivanov	University of Pittsburgh
Douglas Densmore	Boston University
Jenhan Tao	University of California San Diego
Howard Salis	The Pennsylvania State University
Marilene Pavan	Lanzatech
Soha Hassoun	Tufts University

Program

Monday, July 8th

SBOL Workshop (http://sbolstandard.org/iwbda-2019/)

Workshop on Bio-Design for Portability (https://pzuliani.github.io/bd4p.html)

Tuesday, July 9th

08:00 - 09:00 Breakfast and Registration

09:00 - 09:10 **Welcome & Opening Remarks** Pietro Lio' (University of Cambridge)

09:10 - 10:30 **Session I: Design Automation**, Chair: Bradley Brown (Newcastle University)

- DNAWeaver: optimal DNA assembly strategies via supply networks and shortest-path algorithms
  Valentin Zulkower, The Edinburgh Genome Foundry Team, and Susan Rosser
- An automated QC pipeline for libraries with high degree of variants
  Ernst Oberortner, Robert Evans, and Jan-Fang Cheng
- Automated Design and Characterization of Large Toolboxes of Highly Non-Repetitive Genetic Parts
  Ayaan Hossain and Howard M. Salis
- The Operon Refactoring and Construction Assistant (ORCA): Streamlined gene cluster refactoring
  Ernst Oberortner, Nathan J. Hillson, and Jan-Fang Cheng

10:30 - 11:00 **Break**

11:00 - 12:00 **Keynote I: Luca Cardelli (University of Oxford)**

12:00 - 13:30 **Lunch**

13:30 - 14:30 **Session II: Machine Learning**, Chair: Jacob Beal (Raytheon BBN Technologies)

- Towards a framework for implementing evolutionary algorithms in bacterial colonies
  Javier Carrión, Yerko Ortiz, and Martín Gutiérrez
- Combining metabolic modelling with machine learning accurately predicts yeast growth rate
  Christopher Culley, Supreeta Vijayakumar, Guido Zampieri, and Claudio Angione
- Towards Detection of Engineering in Metagenomic Sequencing Data for Yeast and Other Fungi
  Sancar Adali, Aaron Adler, Joel S. Bader, John Grothendieck, Thomas Mitchell, Anton Persikov, Jonathan Prokos, Richard Schwartz, Mona Singh, Allison Taggart, Benjamin Toll, Stavros Tsakalidis, Daniel Wyschogrod, Fusun Yaman, Eric Young, and Nicholas Roehner

14:30 - 15:00 **Break**

15:00 - 15:40 **Session III: Computation in Genetic Circuits**, Chair: Prashant Vaidyanathan (Microsoft)

- Biophysical Analysis for Implementing Neuro-inspired Computing in Living Cells

  Ramez Danial, Luna Rizik, Ximing Li, and Raghd Abu Sinni

- Analyzing Genetic Circuits for Hazards and Glitches

  Pedro Fontanarrosa, Hamid Hosseini, Amin Espah Borujeni, Yuval Dorfan, Chris Voigt, and Chris Myers

15:40 - 16:10 **Poster Pitches**, Chair: Prashant Vaidyanathan (Microsoft)

- A Logic Programming Language for Computational Nucleic Acid Devices

  Carlo Spaccassassi, Matthew R. Lakin, and Andrew Phillips

- CellScanner: a software for extracting physical features from individual cells

  Sebastián Antón, Marco Clavero, and Martín Gutiérrez

- OptBioDes: optimal design for the synbio toolchain

  Pablo Carbonell, Rainer Breitling, Jean-Loup Faulon, and The SYNBIOCHEM Team

- Integration of Performance Metrics into Microfluidic Design Automation

  Radhakrishna Sanka and Douglas Densmore

- Modular Microfluidic Design Automation Using Machine Learning

  Ali Lashkaripour, Christopher Rodriguez, Noushin Mehdipour, David McIntyre, and Douglas Densmore

- Accelerating the Threshold and Timing Analysis of Genetic Logic Circuit Models

  Sanaullah, Hasan Baig, and Jeong A Lee

- Better research by efficient sharing: evaluation of free management platforms for synthetic biology designs

  Uriel Urquiza-García, Tomasz Zieliński, and Andrew J. Millar

- A method for compiling arbitrary transfer functions to molecular circuits

  Iuliia Zarubiieva, Francesca Mantellino, Andrew Phillips, and Vishwesh Kulkarni

- The Morphogen Circuit Builder & Compiler

  Bryan Bartley, Brian Basnight, Jesse Tordoff, Jacob Beal, Ron Weiss

- Machine Learning Algorithms for Robust Meta-Analysis of Gene Expressions

  Vishwesh Kulkarni, Xinwu Yu, and Weikang Qian

- Mutation of synthetic constructs in E. coli

  Duncan Ingram, Mark Isalan, and Guy-Bart Stan

- Parallel Binary Sorting and Shifting with DNA

  Tonglin Chen and Marc Riedel

16:10 - 17:30 **Poster Session**

Wednesday, July 10th

09:00 - 09:10 **Welcome & Opening Remarks**: TBD

09:10 - 10:30 **Session IV: Modelling**, Chair: Nicholas Roehner (Raytheon BBN Technologies)

- An automated model reduction tool to guide the design and analysis of synthetic biological circuits

  Ayush Pandey and Richard M. Murray

- Estimating Biologically Relevant Network Structures from Time-series Data

  Zoltan A. Tuza and Guy-Bart Stan

- Model-driven design of genetic regulatory networks using virtual parts

  Göksel Mısırlı, Bill Yang, and Anil Wipat

- Stochastic Analysis of an Genetic Sensor

  Jeanet Mante, Pedro Fontanarrosa, and Chris Myers

10:30 - 11:00 **Break**

11:00 - 12:00 **Keynote II: Traci Haddock-Angelli (iGEM Foundation)**

12:00 - 13:30 **Lunch**

13:30 - 14:30 **Session V: Data standards and Visualization**, Chair: Ernst Oberortner (Berkeley Lab)

- Visualization of Part Use in SynBioHub

  Jeanet Mante, Zach Zundel, and Chris Myers

- SBOL Visual 2 Ontology

  Göksel Mısırlı, Jacob Beal, Thomas E. Gorochowski, Guy-Bart Stan, Anil Wipat, and Chris Myers

- Flapjack: an open-source tool for storing, visualising, analysing and modelling kinetic gene expression data

  Guillermo Yáñez, Isaac Núñez, Tamara Matute, Fernán Federici, and Timothy J. Rudge

14:30 - 15:00 **Break**

15:00 - 16:00 **Session VI: Microfluidics**, Chair: Radhakrishna Sanka (Boston University)

- A Reconfigurable Digital Microfluidics Platform

  Georgi Tanev, Luca Pezzarossa, Winnie E. Svendsen, and Jan Madsen

- Design Automation of Microfluidic Droplet Sorting Platforms

  David McIntyre and Douglas Densmore

- Detecting Engineering in Single Cells using Tapestri Microfluidics

  Aaron Adler, Adam Abate, Joe Collins, Ben Demaree, Kevin Keating, Xiangpeng Li, Thomas Mitchell, David Ruff, Allison Taggart, Shu Wang, Daniel Weisgerber, Daniel Wyschogrod, Fusun Yaman, Eric M. Young, and Nicholas Roehner

16:00 - 16:30 **Break**

16:30 - 17:10 **Discussion/Special Session: Jacob Beal**

Topic: *"Keeping the Promises of Synthetic Biology."*

17:10 - 17:30 **Wrap up, Announcements, & Award Ceremony**

Keynote Presentation

Integrated Scientific Modelling and Lab Automation

Luca Cardelli

Speaker Biography

Prof. Luca Cardelli is a Royal Society Research Professor at the University of Oxford since 2013. He has an M.Sc. in computer science from the University of Pisa, and a Ph.D. in computer science from the University of Edinburgh. He worked in the USA at Bell Labs, Murray Hill, from 1982 to 1985, and at Digital Equipment Corporation, Systems Research Center in Palo Alto, from 1985 to 1997, and at Microsoft Research, in Cambridge UK from 1997 to 2018 where he was head of the Programming Principles and Tools and Security groups until 2012. His main interests are in programming languages and concurrency, and more recently in programmable biology and nanotechnology. He is a Fellow of the Royal Society, a Fellow of the Association for Computing Machinery, an Elected Member of the Academia Europaea, and an Elected Member of the Association Internationale pour les Technologies Objets.

Keynote Abstract

The cycle of observation, hypothesis formulation, experimentation, and falsification that has driven scientific and technical progress is lately becoming automated in all its separate components. However, integration between these automated components is lacking. Theories are not placed in the same formal context as the(coded) protocols that are supposed to test them: neither description knows about the other, although they both try to describe the same process. We develop integrated descriptions from which we can extract both the model of a phenomenon (for possibly automated mathematical analysis), and the step carried out to test it (for automated execution by lab equipment). This is essential if we want to carry out automated model synthesis, falsification, and inference, by taking into account uncertainties in both the model structure and in the equipment tolerances that may jointly affect the results of experiments.

Keynote Presentation

Setting Standards in Synthetic Biology

Traci Haddock-Angelli

Speaker Biography

Dr. Traci Haddock-Angelli earned her doctorate in marine microbiology from the University of Rhode Island in 2010. She then joined the CIDAR lab under the guidance of Prof. Douglas Densmore at Boston University as a postdoctoral researcher in synthetic biology and later served as the Executive Director of the Center of Synthetic Biology. During her time at Boston University, Traci mentored over 40 undergraduate and graduate students and also ran the university's iGEM team for four years. Traci joined the iGEM Foundation in the spring of 2015 as a Science and Technology Fellow, was later promoted to the Director of Technology, and now serves as the Director of the Competition. In this role, Traci oversees all aspects of the iGEM Competition and works to ensure that every team from around the world has a successful and fulfilling iGEM experience.

Keynote Abstract

Since 2003, the iGEM Foundation has been working on setting standards in synthetic biology. From the early days of BioBricks, we have worked to introduce the idea of engineering standards to young synthetic biologists through the competition and to the rest of the synthetic biology community through the Registry of Standard Biological Parts. Recent work has focused on introducing a new assembly standard to the Registry and on the development of standard protocols to promote the use of absolute units for fluorescence measurements. Through the development of these standards, we know that community involvement is crucial to the creation and adoption of new standards, and invite you all to join us as we work towards creating a strong, responsible, and visionary synthetic biology industry.

Allan Kuchinsky Scholarship

Martín Eduardo Gutiérrez Pescarmona

Dr. Martín Gutiérrez received his B.Sc. in Industrial Engineering and M.Sc. in Computer Science from Pontificia Universidad Católica de Chile in 2005. In 2006, he was appointed as Lecturer at Universidad Diego Portales, Chile, where he taught all branches of Computer Science and specialized in AI techniques and meta-heuristic algorithms. Later, Martín completed his Ph.D. thesis from 2012 to 2017, under the supervision of Prof. Alfonso Rodríguez-Patón at the AI Lab at Universidad Politécnica de Madrid, Spain. During this period, his research was focused on Agent based Model bacterial colony simulators, synthetic gene circuits, bacterial conjugation, and spatio-temporal patterns in bacterial colonies. The main contribution of his thesis was extending the gro simulator. Dr. Gutiérrez briefly continued his work as a Postdoctoral researcher at the AI Lab of Universidad PolitÃĽcnica de Madrid before returning to Universidad Diego Portales in 2018. He is currently the Assistant Professor at Escuela de Informática y Telecomunicaciones, where he continues his research on automation techniques applied to bacterial/cell colony simulators, interplay of AI and synthetic biology, spatio-temporal patterns and inter-cell communication based circuits.

The fifth annual Allan Kuchinsky scholarship is generously sponsored by Agilent.

Previous recipients

2018 - Dr. Steve Shih
2017 - Dr. Curtis Madsen
2016 - Dr. Nicholas Roehner
2015 - Dr. Swapnil Bhatia

Agilent Technologies

Oral Presentations

Poster Presentations

DNA Weaver: optimal DNA assembly strategies via supply networks and shortest-path algorithms

Valentin Zulkower
valentin.zulkower@ed.ac.uk
Edinburgh Genome Foundry
School of Biological sciences
The University of Edinburgh, UK

The Edinburgh Genome Foundry Team
School of Biological Sciences
The University of Edinburgh, UK

Susan Rosser
Edinburgh Genome Foundry
SynthSys Centre
School of Biological Sciences
The University of Edinburgh, UK

1 MOTIVATION

Synthetic Biology applications often require the assembly of large DNA sequences from smaller fragments. This can be a long and expensive process, involving the careful design of many fragments sequences and multiple assembly steps. While software can help alleviate these challenges, existing solutions focus on specific scenarios, such as plasmid assembly from standardized genetic parts [1], and chromosome synthesis from commercially ordered DNA blocks [2] or oligonucleotides [6]. However, assembly projects may use different combinations of cloning methods (e.g. Gibson Assembly, Golden Gate assembly, recombination in yeast) and DNA sources (parts repositories, genomic DNA, commercial providers), depending on the desired DNA sequence and the researcher's preferences.

We present DNA Weaver, an open-source Python framework[1] and web application[2] to compute cost- and time-optimal assembly strategies from user-specified DNA sources and cloning methods. DNA Weaver combines supply-network models with shortest-path algorithms to return either perfect solutions or quick approximations to a variety of problems. This makes it suitable to plan routine cloning in a research laboratory, to automatically evaluate the costs and complexity of projects submitted online to a biofoundry, or to automatically optimize the design of novel sequences towards better manufacturability.

2 METHODS

Problem definition via supply networks

Users define an assembly problem either via Python scripts or the web interface, by providing the desired final sequence and designing a supply network representing all available cloning options (Figure 1A). Each node of the supply network represents a DNA source, e.g. a commercial provider, a parts repository, a PCR station which extracts DNA from existing constructs or genomes, or an assembly station which assembles supplied fragments into a larger one. Each source can be extensively parametrized to reflect its capabilities. For instance, DNA vendors can have custom sequence acceptance rules (see [5] for examples), pricing policies, and lead times. PCR stations can have a range of acceptable oligo primer lengths and annealing temperatures, as well a BLAST database of available constructs and genomes. Assembly stations can implement different cloning methods defined by parameters such as fragments overhangs, restriction sites used, acceptable size ranges of the fragments and final assembly, forbidden sequence patterns, etc.

Each edge of the supply network indicates a client-supplier relationship between two DNA sources. If a PCR station needs a primer oligonucleotide, or an assembly station needs a particular DNA fragment, they request the sequence from all of their direct suppliers. Each supplier indicates whether it can provide the sequence, at which price, and with what lead time. The client station then retains the best offer as the fragment price. This chain of competitive bidding between members of the network ensures that each DNA fragment of each assembly is obtained using the cheapest (and most adapted) vendors and assembly methods.

The variety of possible supply network layouts enables DNA Weaver to solve various cloning problems, from multi-step assembly (Figure 1A) to site-directed mutagenesis or the auto-completion of parts library assemblies with commercially ordered sequences (interactive examples of such scenarios are provided in the web application). In addition, the use of supply networks makes it easy to plug in new classes of DNA sources with different behaviors (which users can define using the Python language), for instance sources connected to remote sequence databases or vendor APIs.

Graph-based sequence decomposition

To provide a sequence at the best price, an assembly station must find the cheapest possible set of fragments which can be assembled into the desired sequence. This often amounts to identifying which regions of the sequence can be inexpensively obtained from DNA repositories or from the cheapest vendors. The sequence decomposition algorithm starts by building a costs graph (Figure 1B) where each edge represents a possible sequence fragment, and an edge's weight is

[1] https://github.com/Edinburgh-Genome-Foundry/DnaWeaver
[2] https://dnaweaver.genomefoundry.org

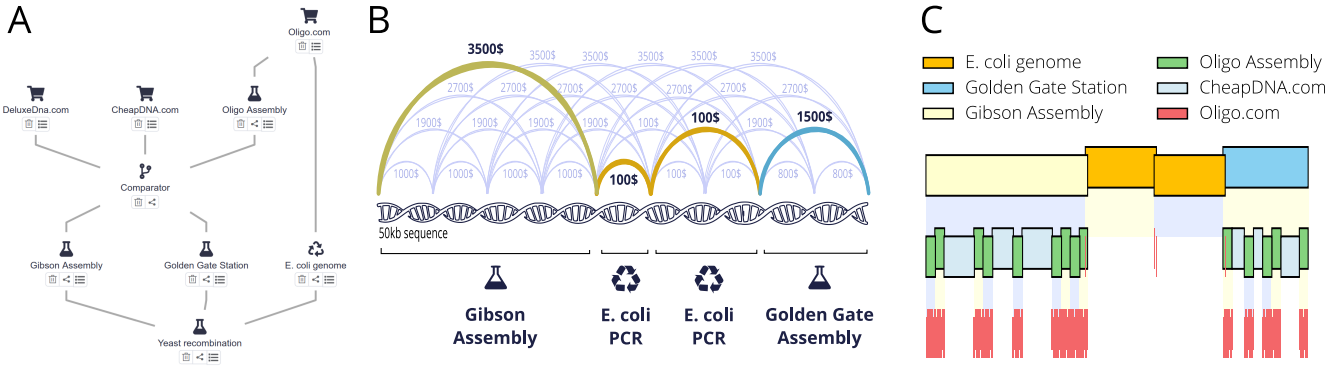


Figure 1: DNA Weaver's web interface, sequence decomposition algorithm, and output. (A) Web interface screenshot of a user-defined supply network modeling a multi-step assembly protocol. The final sequence is assembled via recombination in yeast (bottom station) from large fragments obtained either via Gibson assembly, Golden Gate assembly, or via PCR of the *Escherischia coli* (*E. coli*) chromosome. The Golden Gate and Gibson stations assemble fragments originating either from two commercial providers (cart icons), or from oligonucleotide assembly. (B) Simplified representation of the costs graph used by the yeast recombination station in panel A to decompose a 50kb sequence. The highlighted shortest path indicates the cheapest solution, where lateral fragments are obtained from assembly stations, and the central region's homology to *E. coli* is exploited to obtain cheap DNA fragments via genomic PCR. (C) Assembly plan for the 50kb sequence in panel B, from oligo assembly (bottom level) to the final assembly in yeast (top level), returned in under 10 seconds by the web application.

the fragment's cost (accounting for the addition of assembly overhangs). The shortest graph path from the first nucleotide to the last can be computed using the Dijkstra algorithm [3] and gives the cost-optimal sequence decomposition.

The computational complexity to cost all edges is $O(L^2)$ (where L is the sequence length). DNA Weaver also implements approximate resolution approaches such as the A* path-finding algorithm [3] or nucleotide-skipping, which can reduce the number of edges to compute by orders of magnitude, allowing quasi-real-time sequence editing with manufacturability feedback.

Further graph operations allow to model the limitations of a cloning method, e.g. by limiting the number of parts in an assembly, constraining fragments sizes, forbidding incompatible overhangs to be used in a same cloning step, or preventing high-GC regions to be used as overhangs. It is also possible, by filtering graph edges based on supplier lead time, to obtain the cheapest cloning strategy under a time constraint, or the fastest assembly plan within a given budget.

Output and use for sequence optimization

DNA Weaver generates comprehensive assembly planning reports featuring plots of the final assembly plan (Figure 1C), PDF documents summarizing all sequence ordering assembly steps, and spreadsheets listing all sequences to order.

For design optimization purposes, DNA Weaver can return an analysis of the cost graph pinpointing the sequence locations susceptible to impact assembly cost. This information can then be used to iteratively re-design the desired sequence towards lower costs and manufacturing complexity, possibly taking into account other application-specific objectives and constraints, via existing DNA optimization software such as DNA Chisel [4].

Acknowledgments

The Edinburgh Genome Foundry is supported by the BBSRC (BB/M025659/1, BB/M025640/1, and BB/M00029X/1 to YC) and the BBSRC/MRC/EPSRC funded UK Centre for Mammalian Synthetic Biology (BB/M0101804/1) as part of the RCUK's Synthetic Biology for Growth programme.

REFERENCES

[1] Appleton, E., Tao, J. H., Haddock, T., and Densmore, D. Interactive assembly algorithms for molecular cloning. *Nat Methods 11*, 6 (2014), 657–+.

[2] Christen, M., Del Medico, L., Christen, H., and Christen, B. Genome Partitioner: A web tool for multi-level partitioning of large-scale DNA constructs for synthetic biology applications. *PLOS ONE 12*, 5 (2017), 1–19.

[3] Delling, D., Sanders, P., Schultes, D., and Wagner, D. Engineering route planning algorithms. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2009).

[4] Edinburgh Genome Foundry. DNA Chisel, a generic sequence optimizer. https://github.com/Edinburgh-Genome-Foundry/DnaChisel.

[5] Oberortner, E., Cheng, J. F., Hillson, N. J., and Deutsch, S. Streamlining the Design-to-Build Transition with Build-Optimization Software Tools. *ACS Synthetic Biology* (2017).

[6] Richardson, S. M., Mitchell, L. A., Stracquadanio, G., Yang, K., Dymond, J. S., DiCarlo, J. E., Lee, D., Huang, C. L. V., Chandrasegaran, S., Cai, Y., Boeke, J. D., and Bader, J. S. Design of a synthetic yeast genome. *Science* (2017).

An automated QC pipeline for libraries with high degree of variants

Ernst Oberortner, Robert Evans, Jan-Fang Cheng
DOE Joint Genome Institute (JGI)
{eoberortner,revans,jfcheng}@lbl.gov

INTRODUCTION

The U.S. Department of Energy (DOE) Joint Genome Institute (JGI) provides DNA sequencing and synthesis services to scientific users via community science programs. The DNA synthesis program[1] covers all aspects of the synthetic biology design-build-test cycle, enabling users to engineer biological systems that are relevant to their own research roadmap and the DOE mission.

One particular product type, among others, that the DNA synthesis program offers is the design and construction of libraries with a high degree of variants. Applications of such libraries include genome editing using whole genome guide RNA (gRNA) libraries or studying structural variations of genes or regulatory elements (e.g., promoters, terminators or ribosomal binding sites). At DOE JGI, the workflow for designing and building such libraries include (i) *in silico* design of the library variants depending on its application and its destination vectors, (ii) synthesis of the library variants by commercial DNA synthesis vendors as oligo pools, (iii) amplification of the oligo pool and cloning into its destination vectors, (iv) transformation into *E. coli*, (v) quality control (QC) of the transformed library using high-accuracy, short-read next-generation sequencing (NGS) technologies and, if all QC standards are met, (vi) shipment of the library to the user.

Here, we present our recently established bioinformatics pipeline to QC the transformed libraries. The pipeline comprises freely available, off-the-shelf bioinformatics tools to process the short NGS reads as well as newly developed scripts to visualize the QC results. The QC results depend on (i) the percentage of NGS reads that contain any library variant including the 5‘and 3‘vector overlaps and (ii) the equal representation of all library variants, avoiding bias towards the over- or under-representation of any variant.

RESULTS & DISCUSSION

The main purpose of this QC pipeline (Figure 1) is to detect the representation of each library variant. To do that, we (i) assemble Illumina short paired-end reads, (ii) separate the reads with perfect matches to the expected variants from reads with mis-matches, and (iii) plot the frequency of perfectly matched variants. We normally generate an average of 100-200 folds assembled read coverage of the variants in the library to produce a good coverage plot.

We evaluated the QC pipeline with four user projects, including (i) whole genome gRNA libraries targeted for organisms including *Pseudomonas putida*, *Clostridium autoethanogenum* and *Yarrowia lipolytica* and (ii) partial genome, functional-oriented (eg. metabolic genes) gRNA libraries for *Saccharomyces cerevisiae*. The pipeline is, however, applicable for gRNA libraries for gene suppression, activation or genome editing.

Currently, we have evaluated our QC pipeline using Illumina MiSeq sequencing data only. The pipeline is, however, applicable to any high-accuracy, short-read sequencing technology, such as the Illumina HiSeq or NextSeq systems.

METHODS

Step I: *In silico* design of library variants

For the design of whole genome gRNA libraries, we utilize the CCTop CRISPR/Cas9 target online predictor [2]. The computational design is, however, out of scope in this abstract.

Step II: Oligos synthesis

Oligos of up to 200 bases are commercially available as mixed oligo pools (Twist Biosciences, CA). The level of variants and the sequence fidelity of these oligos are suitable for constructing whole genome gRNA libraries.

Step III: Library assembly and cloning

Although the quantity of the oligos is sufficient for one cloning attempt, a few cycles of PCR amplification is recommended to generate enough materials for a few attempts without introducing significant biases. We have successfully used both Gibson and Golden Gate assembly and cloning approaches to construct whole genome gRNA libraries with over 15,000 variants. A successful library is defined as containing more than 10-fold colony-forming unit (CFU) than the number of variants. The library is amplified once by plating it out on large bioassay plates and harvesting cells in freezing medium. The amplified library can be stored at

[1] https://jgi.doe.gov/user-programs/program-info/csp-overview/synthetic-biology/

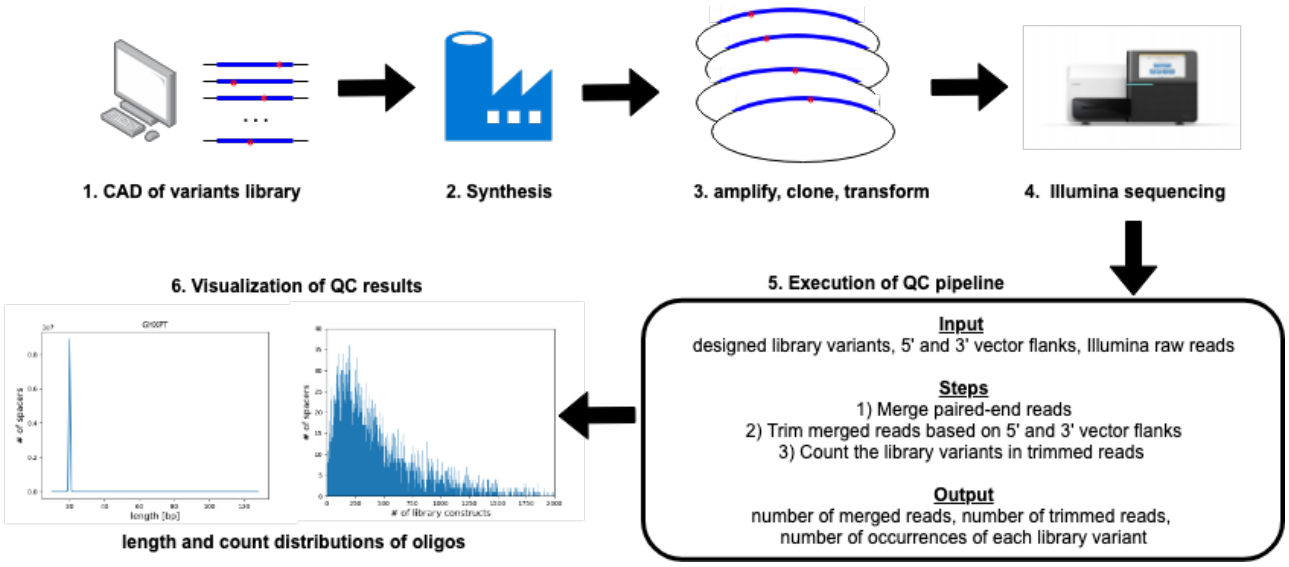


Figure 1: The automated QC pipeline and its integration into the design and build workflows of libraries with a high degree of variants. All steps, except the synthesis step, are performed at DOE JGI.

$-80C$ or used for DNA preparation and introduction into the targeted organism.

Step IV: High-accurracy, short-read sequencing

To facilitate high coverage sequencing of the inserted library, the region of interest is isolated using PCR with minimal cycling or restriction digest followed by size-selection. The isolated DNA is then sequenced using an Illumina MiSeq system.

Step V: Automated QC pipeline

Our pipeline mainly consists of individual bioinformatics tools, which are available via BBTools[2]. The user provides two inputs: (i) a FASTQ file, containing the overlapping paired-end reads generated by the Illumina MiSeq sequencer and (ii) a spreadsheet that contains the library variants including their 5‘and 3‘vector flanking sequences.

First, we use `bbmerge` [1] to merge the paired-end Illumina reads. Then, we use `seal`[3] to filter the merged reads based on the 5‘and 3‘vector overlaps, whose positive and negative strand sequences are encoded in a FASTA file. Then, we use `bbduk`[4] to trim the 5‘and 3‘vector overlaps from each filtered read. The resulting FASTQ file contains the filtered and trimmed reads that should contain and match the designed library variants. This FASTQ file serves then as input to `bbduk` and `seal` to respectively generate a textual representation of a length distribution histogram of the trimmed reads and a count summary of how often each library variant occurs in the trimmed reads.

Step VI: Visualization of QC results

The final step of our QC pipeline is the visualization of the textual outputs of `bbduk` and `seal`. Therefore, we developed a Python script that imports the length distribution histogram and the count summary. Our Python script uses Matplotlib, python's 2D plotting library, to generate the plots for visualizing the QC results, helping the current manual decision making process to either ship the library to the scientific user or to perform any rework.

ACKNOWLEDGMENTS

We thank Brian Bushnell for his support on utilizing various BBTools to establish our pipeline. The computational resources for hosting and executing the pipeline is provided by DOE National Energy Research Scientific Computing Center (NERSC). This work has been conducted by the U.S. Department of Energy Joint Genome Institute, a DOE Office of Science User Facility, is supported under Contract No. DE-AC02-05CH11231.

[2]https://jgi.doe.gov/data-and-tools/bbtools/
[3]https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/seal-guide/
[4]https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/

REFERENCES

[1] Bushnell et al. 2017. BBMerge — Accurate paired shotgun read merging via overlap. *PLOS ONE* 12 (10 2017), 1—15. https://doi.org/10.1371/journal.pone.0185056
[2] Stemmer et al. 2015. CCTop: An Intuitive, Flexible and Reliable CRISPR/Cas9 Target Prediction Tool. *PLOS ONE* 10 (04 2015), 1—11.

Automated Design and Characterization of Large Toolboxes of Highly Non-Repetitive Genetic Parts

Ayaan Hossain
auh57@psu.edu
Pennsylvania State University
University Park, Pennsylvania, USA

Howard M. Salis
salis@psu.edu
Pennsylvania State University
University Park, Pennsylvania, USA

1 SYNTHETIC BIOLOGY REPEAT CHALLENGE

Repetitive design is ubiquitous in modular engineering. Synthetic biologists have previously embraced this design philosophy in engineering multi-input-multi-output logics [3], signal processing [2] and feedback control [4] - often repeating genetic parts (including genes) with desired activity multiple times in a system. This approach simplifies the underlying physics by eliminating part variability, but, the resulting systems become repetitive, making them difficult to synthesize using commercially available DNA synthesis approaches, and unstable in host organisms that exhibit homologous recombination activity. 30-bp repeats in a long DNA construct can cause complete synthesis failure or exorbitant delays 80% of the time. In popular engineering chassis such as *E. coli* and *S. cerevisiae* 15 to 20-bp repeats are sufficient in triggering detectable recombination, during which sections of the system flanked by repeats are deleted, and the system becomes functionally compromised. Collectively, these problems constitute the *synthetic biology repeat challenge*. Previously, synthetic biologists have overcome this challenge by manually mutagenizing parts [2, 3], or switching to a different genetic part potentially compromising system efficiency. Both approaches are sub-optimal and trial-and-error based. If we were to engineer large and stable genetic systems, pathways, and refactor whole genomes, we would need efficient approaches to design and characterize very large toolboxes of different types of non-repetitive genetic parts that are presently absent in literature (see Table 1).

	Available Parts	Non-Repetitive
E. coli Promoters	11,049	13
E. coli Terminators	647	154
S. cerevisiae Promoters	23	10
S. cerevisiae 5' UTRs	5,032	104
S. cerevisiae 3' UTRs	66	14

Table 1: Availability of non-repetitive ($L_{max} = 10$) genetic parts in some commonly used toolboxes.

2 THE NON-REPETITIVE PARTS CALCULATOR

To engineer large toolboxes of highly non-repetitive genetic parts, we developed the Non-Repetitive Parts Calculator (see Figure 1). Given the highly experimental nature of synthetic biology, we propose two complementary modalities within NRP Calculator - the Finder Mode and the Maker Mode. Formally, the Finder Mode algorithm is a top-down discovery approach in which we assume that a synthetic biologist has access to a large toolbox of characterized genetic parts, from which she wants to select a large subset of parts, such that no two parts in the resulting toolbox share repeats of length $\geq L_{max}$, the maximum allowed repeat length. We denote such a collection of genetic parts as a Non-Repetitive Toolbox. To solve this problem, we formulate the initial toolbox as a *repeat graph* and connect every pair of vertex with an edge if their labelled parts share repeats $\geq L_{max}$. We then eliminate an optimally small vertex cover of the graph, so that the independent set of the graph encodes a Non-Repetitive Toolbox. In contrast to Finder Mode, the Maker Mode algorithm is a bottom-up design approach. We

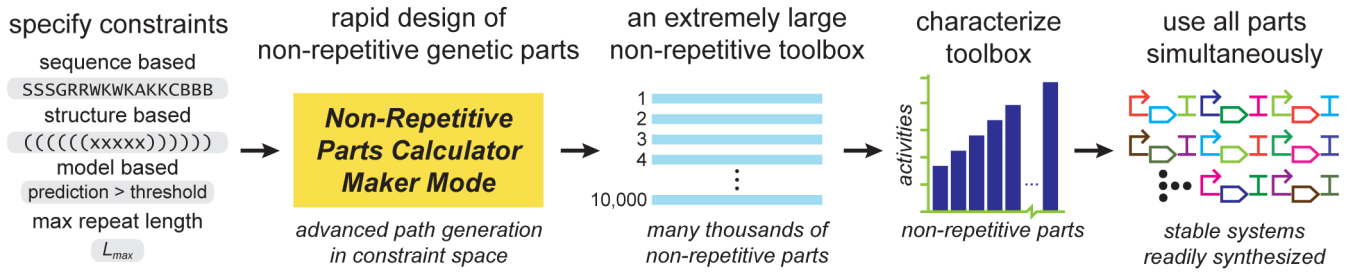


Figure 1: NRP Calculator Workflow. A set of design constraints are input to NRP Calculator which generates non-repetitive paths in the defined constraint space to produce large toolboxes of genetic parts fulfilling the design criteria. The toolboxes may then be synthesized, characterized and used simultaneously.

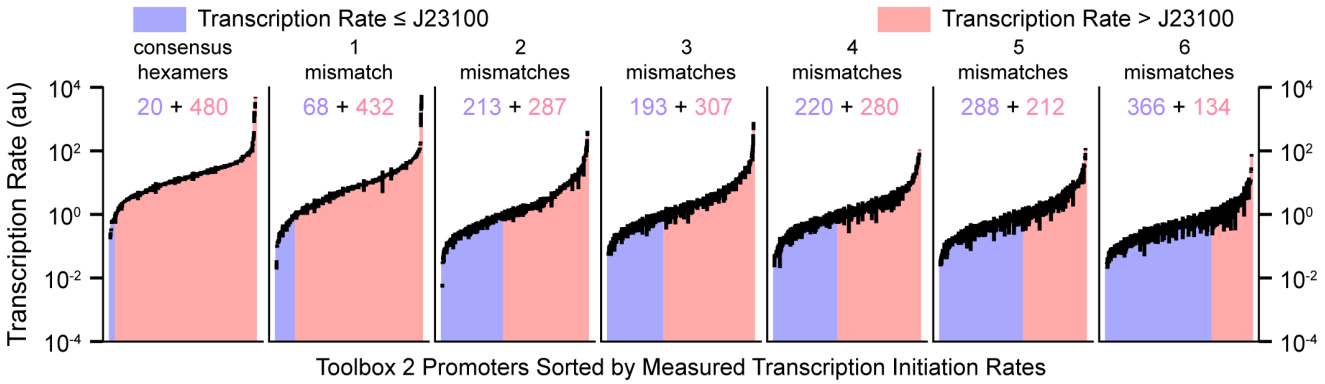


Figure 2: Measured transcription rates from a subset of 3,500 non-repetitive σ^{70} promoters with variable motifs. Black bars represent 1 standard deviation from mean (n = 2).

assume that a synthetic biologist wants to design and then characterize a large toolbox of non-repetitive parts, subject to a set of constraints that ensure the designed parts are functional. Such design constraints may be an IUPAC degenerate nucleotide or IUPAC amino acid sequence constraint, a proscribed RNA secondary structure, and/or a quantitative model of part function, along with L_{max}, which sufficiently encode non-repetitive genetic parts with predictable properties. Given the constraints as input, the Maker Mode algorithm models the design of non-repetitive parts as a path finding problem in constraint space, and outputs either a target sized or maximally largest Non-Repetitive Toolbox. As an analogy to Finder Mode, the Maker Mode algorithm iteratively generates a single non-repetitive part and adds it to the recurring Non-Repetitive Toolbox such that with every addition of a part, the repeat graph corresponding to the recurring toolbox remains disconnected by design. Performance analysis of NRP Calculator showed that both Finder Mode and Maker Mode operated in linear time.

3 NON-REPETITIVE PROMOTER TOOLBOXES FOR *E. COLI* AND *S. CEREVISIAE*

To demonstrate NRP Calculator, we designed and characterized thousands of non-repetitive promoters for developing evolutionarily robust genetic systems composed of hundreds to thousands of non-repetitive parts in *E. coli* and *S. cerevisiae*, which are popularly used organisms for numerous biotechnology applications which often have precise gene regulation requirements. We first designed 4,350 highly non-repetitive (L_{max} = 10) constitutive σ^{70} promoters for *E. coli* up to 3,550 of which may be used simultaneously without introducing any 11-bp repeats. We then used a massively parallel reporter assay to construct and characterize the promoters - combining barcoding, oligopool synthesis, library-based cloning, and next generation sequencing (DNA-Seq and RNA-Seq) to measure their transcription rates. Collectively, our library spanned a 1.15-million-fold in transcription rate dynamic range with the transcription rates being inversely proportional to the mutations in the promoter motifs (see Figure 2). Similarly, we engineered a toolbox of 1,931 non-repetitive (L_{max} = 15) Pol II promoters for concurrent use in *S. cerevisiae*, using Rap1, Reb1 and Gcr1 transcription factor binding sites, and optimized them for reduced nucleosome occupancy thus maximizing RNAP recruitment[1, 5]. Additionally, we optimized the Pol II promoters to be orthogonal to the S288C strain genome to enable efficient recombineering. Characterization in four different media condition showed that the Pol II promoter toolbox spanned at least 1.1-million-fold in transcription rate in synthetic complete media. Post characterization, we explored combinatorial use cases, studied statistical associations and sequence determinants of promoter functionality in order to inform better design constraints, for future iterations. Finally, we showed that non-repetitive design in general not only allowed for composing evolutionarily robust genetic systems, but also enabled robust and parallel sequence assembly, sequence-to-function modelling, and efficient analysis of characterization data by maximizing information content.

REFERENCES

[1] Curran, K. A., Crook, N. C., Karim, A. S., Gupta, A., Wagman, A. M., and Alper, H. S. Design of synthetic yeast promoters via tuning of nucleosome architecture. *Nature Communications 5* (2014), 4002.

[2] Fernandez-Rodriguez, J., Moser, F., Song, M., and Voigt, C. A. Engineering rgb color vision into escherichia coli. *Nature Chemical Biology 13*, 7 (2017), 706.

[3] Nielsen, A. A., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. Genetic circuit design automation. *Science 352*, 6281 (2016), aac7341.

[4] Temme, K., Zhao, D., and Voigt, C. A. Refactoring the nitrogen fixation gene cluster from klebsiella oxytoca. *Proceedings of the National Academy of Sciences 109*, 18 (2012), 7085–7090.

[5] Xi, L., Fondufe-Mittendorf, Y., Xia, L., Flatow, J., Widom, J., and Wang, J.-P. Predicting nucleosome positioning using a duration hidden markov model. *BMC Bioinformatics 11*, 1 (2010), 346.

The Operon Refactoring and Construction Assistant (ORCA): Streamlined gene cluster refactoring

Ernst Oberortner[1], **Nathan J. Hillson**[1,2], **Jan-Fang Cheng**[1]
[1]DOE Joint Genome Institute, [2] DOE Joint BioEnergy Institute
{eoberortner,njhillson,jfcheng}@lbl.gov

INTRODUCTION

The combination of Next-Generation Sequencing (NGS) and bioinformatics enables scientists to discover gene clusters that collectively produce chemical compounds, providing access to a novel, extended diversity of natural products for agriculture, fuels, materials, and medicine. A synthetic biology approach to enhance the biological production of industrially-relevant compounds is to refactor the wild-type gene cluster for expression in a heterologous host. Easily accessible DNA synthesis and assembly services, for example provided by commercial DNA synthesis vendors or biofoundries, enable the manufacturing of the refactored gene clusters in a time- and cost-efficient manner.

The design of a refactored gene cluster comprises of various tasks, such as (i) specifying the pathway and operon structure to modulate the expression-levels of the genes, (ii) assigning or calculating the sequences of the regulatory elements (e.g., promoters, ribosomal binding sites (RBS), terminators), (iii) optimizing the codon usage of genes for expression in the target host, or (iv) preserving or avoiding any sequence motifs (e.g., restriction sites, methylation motifs, internal priming sites) that impact construction and expression.

We have established the Operon Refactoring and Construction Assistant (ORCA), which integrates existing and newly developed biological computer-aided design (bioCAD) tools to enable a fully automated design process for gene cluster refactoring. In a web-based UI, the user inputs data and the values of design parameters according to what design tasks need to be executed in the specific gene cluster refactoring process. A workflow engine then executes the design process according to the user inputs. ORCA outputs the refactored pathway sequences including annotations, ready-to-order sequences for each pathway including instructions about their assembly and cloning into the designed pathway constructs. The value added of using ORCA is, if the user wants, the permanent consideration of synthesis limitations (e.g., %GC content, repeats, sequence motifs) throughout the automated design process.

RESULTS & DISCUSSION

At the U.S. Department of Energy (DOE) Joint Genome Institute (JGI), we developed ORCA according to the design and construction requirements discovered from several user projects, such as refactoring gene clusters for enhanced biological production of dipeptides (cyclic dipeptides, MAAs, and RIPPs), iron-sulfur enzymes, and carboxylic acids. In the dipeptides project, for example, we mined the Integrated Microbial Genomes (IMG) database (https://img.jgi.doe.gov/) and selected an initial set of genes for 20 bacterial pathways for design and construction.

Currently, ORCA supports the design of a single pathway, enabling, however, the design of pathway variants with varying operon structures (see 1). In the near future, we will enable ORCA for batches of pathways and combinatorial designs by integrating bioCAD tools, depending on their applicability, availability, maturity and, moreover, facility for integration (e.g., the JBEI DIVA platform, Eugene [1] or the approach described in [3]).

METHODS

STEP I: Input of genetic parts

The user uploads all parts using a spreadsheet that contains, at a minimum, information about the part's name, type, and sequence. The sequences can also be specified in common sequence file formats (e.g., Genbank, FASTA). In this case, the user also needs to specify name of the sequence file and the part's source location in the referenced file. The ORCA back-end parses the input data and verifies its correctness, such as checking that all referenced files are uploaded and contain the specified locations.

Step II: Design of pathways and operon structures

ORCA supports the following parameters for designing pathways and operon structures: (i) the minimum and maximum number of genes per operon, which determine the number of possible operon configurations, (ii) the orientation of each operon, (iii) the maximum length of an operon in base pairs, and (iv) the number of pathway variants that ORCA should enumerate. In Figure 1, we provide a screenshot of the ORCA web UI. To visualize the pathway designs, ORCA integrates the DNAplotlib library [6], which uses standardized SBOL Visual glyphs [4].

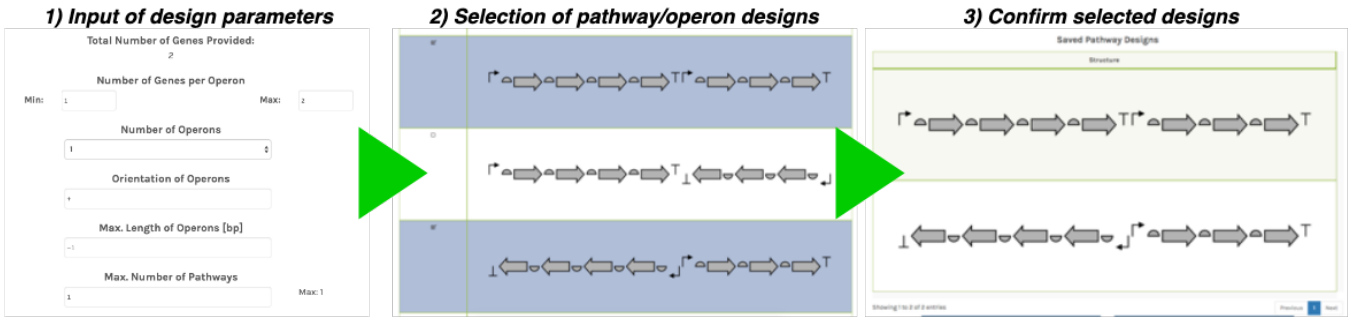


Figure 1: Designing pathways and their operon structures in ORCA

Step III: Selection of promoters and terminators

The user can determine what promoter and terminator sequences should be assigned to each operon of each pathway variant. ORCA provides the following options: (i) select any uploaded promoter and terminator (**Step I**), (ii) assign any promoter and terminator automatically (e.g., to ensure minimal violations against synthesis criteria), or (iii) assign no promoter or terminator to the operon (e.g., when using bidirectional promoters).

Step IV: Calculation of RBS sequences

ORCA integrates existing tools for calculating calculate RBS sequences [2, 7, 8]. Regardless of what tool the user selects, the user can specify how many RBS sequences should be calculated for each gene. Depending on the selected RBS calculation tool, the user needs to provide the corresponding inputs. ORCA also supports uploading of RBS sequences (**Step I**) and to assign them to the genes in the designed pathways (**Step III**).

Step V: Codon-optimization of genes for target host

ORCA integrates the Build-Optimization Software Tools (BOOST) [9], which allow (i) uploading codon usage tables or selection among predefined ones and (ii) selecting among various codon-optimization algorithms.

Step VI: Selection of synthesis criteria and sequence motifs

The user can select among a list of commercial DNA synthesis vendors or manually specify synthesis constraints (%GC, repeats). Throughout the automated design process, ORCA utilizes BOOST to ensure that the pathway sequences contain the minimum number of synthesis constraint violations of either the selected vendor or the user-specified constraints. Similarly as for synthesis constraints, the user can specify sequence motifs that need to be preserved or avoided during the automated design of the pathway sequences.

Step VII: Specification of build instructions

For this task, ORCA again utilizes BOOST, enabling the user to specify build instructions (i.e., synthesis, assembly, and cloning) and to encode them in the Synthetic Biology Open Language (SBOL) and its W3C Provenance extension [5].

ACKNOWLEDGMENTS

We thank Samuel Deutsch, Michalis Hadjithomas and Michael Wornow for guidance, evaluation and initial development efforts. This work has been conducted by the U.S. Department of Energy Joint Genome Institute, a DOE Office of Science User Facility, is supported under Contract No. DE-AC02-05CH11231.

REFERENCES

[1] Bilitchenko et al. 2011. Eugene — A Domain Specific Language for Specifying and Constraining Synthetic Biological Parts, Devices, and Systems. *PLOS ONE* 6 (04 2011). https://doi.org/10.1371/journal.pone.0018882

[2] Bonde et al. 2016. Predictable tuning of protein expression in bacteria. *Nature Methods* 13 (Jan 2016), 233. https://doi.org/10.1038/nmeth.3727

[3] Bhatia et al. 2017. Genetic Design via Combinatorial Constraint Specification. *ACS Synthetic Biology* 6, 11 (2017), 2130—2135. https://doi.org/10.1021/acssynbio.7b00154

[4] Cox et al. 2018. . Vol. 15. Chapter Synthetic Biology Open Language Visual (SBOL Visual) Version 2.0. https://doi.org/10.1515/jib-2017-0074

[5] Cox et al. 2018. . Vol. 15. Chapter Synthetic Biology Open Language (SBOL) Version 2.2.0. https://doi.org/10.1515/jib-2018-0001

[6] Der et al. 2017. DNAplotlib: Programmable Visualization of Genetic Designs and Associated Data. *ACS Synthetic Biology* 6, 7 (2017), 1115—1119. https://doi.org/10.1021/acssynbio.6b00252

[7] Espah Borujeni et al. 2014. Translation rate is controlled by coupled trade-offs between site accessibility, selective RNA unfolding and sliding at upstream standby sites. *Nucleic acids research* 42, 4 (Feb 2014), 2646—2659. https://doi.org/10.1093/nar/gkt1139

[8] Espah Borujeni et al. 2016. Translation Initiation is Controlled by RNA Folding Kinetics via a Ribosome Drafting Mechanism. *Journal of the American Chemical Society* 138, 22 (2016), 7016—7023. https://doi.org/10.1021/jacs.6b01453

[9] Oberortner et al. 2017. Streamlining the Design-to-Build Transition with Build-Optimization Software Tools. *ACS Synthetic Biology* 6, 3 (2017), 485–496. https://doi.org/10.1021/acssynbio.6b00200

Towards a framework for implementing evolutionary algorithms in bacterial colonies

Javier Carrión [1], **Yerko Ortiz**[1], **Martín Gutiérrez**[1]
[1]Escuela de Informática y Telecomunicaciones - Universidad Diego Portales
Santiago, Chile
{javier.carrion,yerko.ortizm,martin.gutierrez}@mail.udp.cl

1 INTRODUCTION

Evolution is a trademark process in biology. Organisms adapt to an environment surrounding them. This concept has been studied and used in Computer Science in the form of algorithms. Furthermore, other higher-level biology processes have also been adapted, and are known as bio-inspired algorithms. Related to this class of algorithms are evolutionary algorithms. The latter rely on the repeated evaluation of multiple possible solutions and follow given heuristics inspired on biological situations to evolve. Oddly enough and to the best of our knowledge, within the advent of Synthetic Biology, these algorithms have been very poorly explored [2, 5]. Massive parallelism, and the dynamics that bacterial colonies follow to evolve and grow offer a new possible platform to implement simulations of these algorithms as a means of achieving an alternative computational paradigm.

2 EVOLUTIONARY ALGORITHMS

Exploration and exploitation mechanisms in search and optimization algorithms are achieved through evolution applied to potential solutions. These solutions are repeatedly evaluated through a fitness function, to evolve towards good ones. Two types of such algorithms are introduced below:

Simulated Annealing (SA): The algorithm is based on a metallurgy annealing process, where the goal is to achieve the best possible crystallization. This is done through controlled decrease of metal temperature, resulting in increasing stability of free energy. In algorithmic terms, one solution is chosen and constantly evaluated within a solution space. The fitness function maps to the energy in the system and mutations are applied to iteratively evolve the solution according to a gradually decreasing temperature label. The algorithm keeps track of the best found solution.

Simple Genetic Algorithm (SGA): This algorithm finds its inspiration in natural selection, where the fittest individuals in a population survive and reproduce. Computationally, this is modeled as a pool of evolving individuals being iteratively evaluated and selected as to persist or to be eliminated. In this context, reproduction is carried out by probabilistically combining existing solutions and then applying mutations on the resulting individuals. The fitter the individual, the greater the probability of surviving and reproducing.

3 PROPOSAL AND MOTIVATION

We propose a framework architecture and implementation to aid in the automation of the generation of evolutionary algorithm simulations in `gro` [3, 4]. We believe the framework concept can be extrapolated to the wet-lab context, and that the associations between operations hold in actual experiments. In addition, we believe evolutionary algorithms implemented in bacterial colonies can coexist with Directed Evolution [1], and be a complement by providing formal and structured guidelines to configure the process. Another application of this framework could be towards the construction and evaluation of evolvable synthetic circuits. Taking advantage of the versatility in programming cells along with the native environment involving evolution and the intrinsic parallelism occurring in cell colonies provides a favorable environment for implementation. A first approach in this direction can be to design and implement a tool that generates `gro` simulation templates for evolutionary algorithms in bacterial colonies.

4 DESIGN AND IMPLEMENTATION

The framework includes the high-level definition of evolutionary algorithms. Also, it is based on a proposed mapping from typical algorithm operations to biological processes used in Synthetic Biology. This mapping establishes general recommendations for the implementation of experiments/simulations of evolutionary algorithms in bacterial colonies. Both of these elements are at the core of the framework. Also, a proof-of-concept was implemented to test the behavior of the algorithms. This is, a generator for `gro` simulation specification files of the described evolutionary algorithms was constructed to ease and accelerate the process. The generator captures the general algorithm logic through the proposed mappings, and together with the input parameters, constructs a template `.gro` specification file.

Toolkit

At this time, the toolkit includes the definition of SGA and SA templates. Operations such as mutation and natural selection

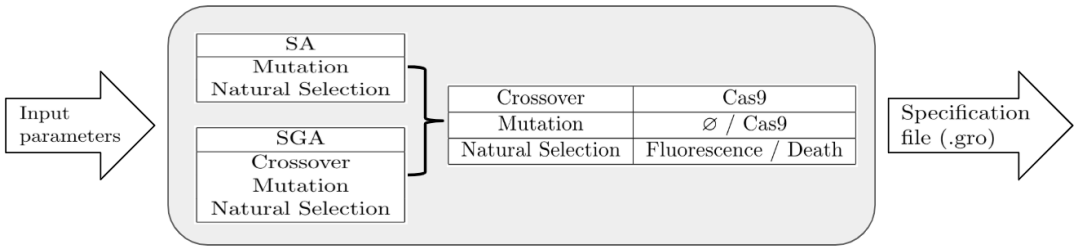


Figure 1: Entire simulation generation process. The user selected parameters and choices integrate into a `.gro` file that acts as a template. On the other hand, each algorithm representation includes its computational operations. These operations are then related to biological processes. Cas9 refers to a protein that triggers sequence editing of genes.

have been conceptually generalized to fit their respective roles in each of the algorithms. Furthermore, the operations are mapped to a proposed set of biological processes that can represent them in an experimental environment. This toolkit has been designed to be extensible in algorithms and operations. A summary of the relationship between algorithm operations and biological processes can be found in Figure 1.

Specification file generator

A tool that generates output `gro` specification template files for evolutionary algorithms was implemented. This generator requires input parameters chromosome length and the representation of a fitness function in terms of presence or absence of each component (protein) to operate and configure the template. Additionally, the user must select the evolutionary algorithm to generate. Specific functions such as crossover and mutation must be specified by the end user. However, they are considered within the template.

5 RESULTS

Our simulations are aimed at finding bacteria expressing GFP proteins: bacteria tagged as good solutions. In both SA and SGA, this is a boolean function imposing the presence and/or absence of all control proteins, e.g.: the fitness function for a bacterium could be the presence of two specific proteins and the absence of other two proteins, all four of them different. The parameter modified for each test is the conjugation rate. In the case of SA, it represents the mutation rate of the algorithm, as opposed to SGA, where it represents the crossover rate. Tables 1 and 2 summarize the mean values over 10 simulations of each kind: data for number of GFP expressing bacteria, total number of bacteria and number of generations of the first occurrence of GFP bacterium were recorded. Source code can be found at *https://github.com/jcarrionramos/Bioinspired-framework*.

Table 1: SA algorithm runs varying mutation probability

P_m	GFP	Total	Number of Generations
0.2	16	9986	2.2
0.1	6	9986	3
0.01	0	9985	N/A

Table 2: SGA algorithm varying crossover probability

P_c	GFP	Total	Number of Generations
0.1	7	9948	7.6
0.01	1	9946	11.6
0.001	0	9946	N/A

6 CONCLUSION

With this framework, we try to approach scientists who are not experts in programming, so they can more easily create evolutionary algorithm simulations in `gro`. This idea was condensed into a tool that generates templates for such algorithms. Preliminary results have been shown from the simulations, demonstrating the functionality of SA and SGA algorithms. As future work, an interesting direction is to implement more evolutionary algorithms within the toolkit. Another possible extension is to link other tools to the framework to aid in the design of initial circuits. All of the mentioned algorithms are currently being validated against their `C++` source code counterparts.

REFERENCES

[1] Arnold, F. H. Design by directed evolution. *Accounts of chemical research 31*, 3 (1998), 125–131.

[2] Beneš, D., Sosík, P., and Rodríguez-Patón, A. An autonomous in vivo dual selection protocol for boolean genetic circuits. *Artificial life 21*, 2 (2015), 247–260.

[3] Gutierrez, M., Gregorio-Godoy, P., Perez del Pulgar, G., Muñoz, L. E., Sáez, S., and Rodríguez-Patón, A. A new improved and extended version of the multicell bacterial simulator gro. *ACS synthetic biology 6*, 8 (2017), 1496–1508.

[4] Jang, S. S., Oishi, K. T., Egbert, R. G., and Klavins, E. Specification and simulation of synthetic multicelled behaviors. *ACS Synthetic Biology 1*, 8 (2012), 365–374.

[5] Rolanía, D. B., Font, J. M., and Manrique, D. Bacterially inspired evolution of intelligent systems under constantly changing environments. *Soft Computing 19*, 4 (2015), 1071–1083.

Combining metabolic modelling with machine learning accurately predicts yeast growth rate

Extended Abstract*

Christopher Culley[1], **Supreeta Vijayakumar**[2], **Guido Zampieri**[2], **Claudio Angione**[2,3]
[1]School of Electronics and Computer Science, University of Southampton, Southampton, UK
[2]Department of Computer Science and Information Systems, Teesside University, Middlesbrough, UK
[3]Healthcare Innovation Centre, Teesside University, Middlesbrough, UK
cc2u18@soton.ac.uk,{g.zampieri,s.vijayakumar,c.angione}@tees.ac.uk

ABSTRACT

New metabolic engineering techniques hold great potential for a range of bio-industrial applications. However, their practical use is hindered by the huge number of possible modifications, especially in eukaryotic organisms. To address this challenge, we present a methodology combining genome-scale metabolic modelling and machine learning to precisely predict cellular phenotypes starting from gene expression readouts. Our methodology enables the identification of candidate genetic manipulations that maximise a desired output – potentially reducing the number of *in vitro* experiments otherwise required. We apply and validate this methodology to a screen of 1,143 *Saccharomyces cerevisiae* knockout strains. Within the proposed framework, we compare different combinations of feature selection and supervised machine/deep learning approaches to identify the most effective model.

KEYWORDS

Genome-scale modelling, machine learning, deep learning, multi-omics, cellular growth, *Saccharomyces cerevisiae*.

1 INTRODUCTION

Cellular growth and gene expression are closely related in unicellular organisms, as they co-participate in mutual regulation. This relationship has yet to be fully understood, and in general predicting cellular growth following genetic manipulations is still challenging. Understanding and controlling cellular growth has important applications in biotechnology for the development of efficient cell factories, but the identification of such strains is still a complex issue [10].

We propose a novel multi-view learning framework that utilises both transcriptomics data and strain-specific metabolic fluxes to predict outputs of bio-industrial interest. To demonstrate the efficacy of this framework, we target it to predicting cellular growth of *S. cerevisiae*, one of the main eukaryotic platforms for bio-industrial production.

*Oral presentation

2 METHODS

In this work, we started from 1,143 *S. cerevisiae* gene expression (GE) profiles – our first data view – each of which are sampled from single deletion strains and are coupled with their corresponding growth rate fold change [8]. We used a genome-scale metabolic model (GSMM) of yeast metabolism [6] in conjunction with METRADE [1] – which uses gene expression to tailor reaction rate bounds – to build an equal number of strain-specific GSMMs. We next used regularised flux balance analysis (RFBA) [11] to determine reaction fluxes for the entire network by maximizing the biomass accumulation rate subject to regulatory and biochemical constraints. The solutions provide steady-state reaction rates (fluxes) for each yeast strain and every reaction in the GSMM. We used the metabolic fluxes (MF) generated in this phase as a second data view in the following prediction stage.

In the supervised learning phase, we employed the following methods: (i) support vector regression (SVR) [3]; (ii) random forest (RF) [4]; and (iii) deep neural networks (DNN). These were selected based on their suitability to build predictive models starting from high-dimensional dataset such as our transcriptomic and fluxomic profiles. We used the caret R package for SVR and RF [9], while DNN were implemented through the keras Python library [5].

Given the high dimensionality of our data, we explored whether feature selection can identify relevant genes or metabolic reactions, to build simpler and more interpretable models. We focused on three state-of-the-art techniques previously applied to omics data: (i) sparse group lasso (SGL) [12]; (ii) non-dominated sorting genetic algorithm II (NSGA-II) [7]; and (iii) iterative random forests (iRF) [2].

3 RESULTS

We developed and evaluated a computational pipeline for predicting *S. cerevisiae* growth rate from experimental and simulated omics data, which is summarised in Figure 1a. In brief, we used strain-specific GSMMs and RFBA to estimate the MF activity of 1,143 yeast mutants in log phase, starting from their GE profiles and optimising the GSMM building

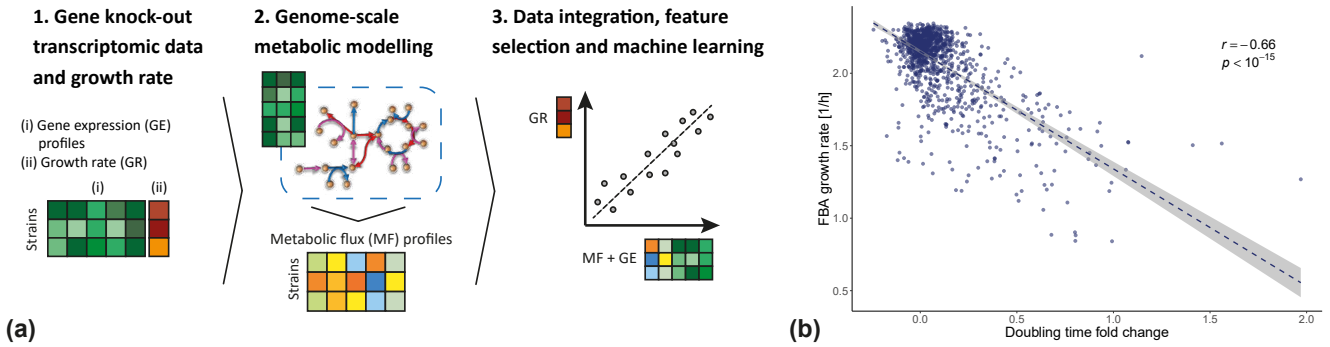


Figure 1: (a) Workflow of the proposed methodology: starting from GE profiles for different synthetic yeast strains – where the colour tones represent rates of GE (greens), target doubling time (reds) and MF (blue to orange) – we build strain-specific GSMMs from which we estimate the MF activity. Next, we build data-driven predictive models using both GE and MF information. (b) Correlation between the growth rate simulated by the strain-specific GSMMs and the relative doubling time for the same strains. This shows that the strain-specific GSMMs correctly capture the metabolic state across strains.

Table 1: Full set of accuracy scores across all dataset-method combinations tested against an unseen set of strains to determine model generalisation: mean absolute error (MAE), median absolute error (MDAE), Pearson's correlation coefficient (PCC). The final column indicates the percentage of fluxomic features (FF%) of the dataset. Here MF+GE corresponds to the full data profiles from both the gene expression and metabolic fluxes.

Dataset	Method	MAE	MDAE	PCC	FF%
MF+GE	SVR	0.080	0.054	0.845	36
MF+GE	RF	0.077	0.048	0.867	36
MF+GE	DNN	0.072	0.049	0.887	36
iRF data	SVR	**0.070**	0.048	0.886	0
iRF data	RF	0.075	0.052	0.869	0
iRF data	DNN	0.073	0.048	0.882	0
NSGA-II data	SVR	0.072	0.049	**0.889**	24
NSGA-II data	RF	0.078	**0.047**	0.843	24
NSGA-II data	DNN	0.081	0.053	0.861	24
SGL data	SVR	0.081	0.057	0.865	34
SGL data	RF	0.081	0.052	0.846	34
SGL data	DNN	0.084	0.058	0.866	34

based on the simulated growth rate (Figure 1b). Then, we built and cross-compared machine and deep learning models predicting yeast growth from integrated GE and MF information (MF+GE), with and without feature selection. In this phase, we tested SVR, RF and DNN in combination with SGL, NSGA-II and iRF. We thereby created three further datasets (SGL data, NSGA-II data and iRF data respectively) comprising the features identified by each of these approaches.

Depending on the combination of dataset and learning algorithm, we observed different trends in prediction scores. Overall, the best performing methods are SVR combined with iRF and NSGA-II, and DNN without prior feature selection. We note that in the case of SVR, feature selection can sensibly improve its prediction accuracy, while there is an opposite trend for DNN. This could suggest that effective DNN models embed non-linear relationships among genes and metabolic reactions that involve a larger set of features. Importantly, the MF variables selected allow us to mechanistically understand the factors governing cell growth and further inform potential manipulations.

4 CONCLUSIONS

Our integrative models enable the joint analysis of experimental genetic regulation patterns and knowledge-based metabolic information to predict yeast cell growth. Our results suggest that integrating multi-omics variables and metabolic modelling can improve yeast growth predictions and provide mechanistic biomarkers. Finally, our pipeline has potential applications in metabolic engineering scenarios, and can be readily extended to other hosts.

REFERENCES

[1] Claudio Angione and Pietro Lió. 2015. *Scientific reports* 5 (2015), 15147.
[2] Sumanta Basu et al. 2018. *Proceedings of the National Academy of Sciences* (2018), 201711236.
[3] Asa Ben-Hur et al. 2008. *PLoS computational biology* 4, 10 (2008), e1000173.
[4] Xi Chen and Hemant Ishwaran. 2012. *Genomics* 99, 6 (2012), 323–329.
[5] François Chollet et al. 2015. https://keras.io.
[6] Ratul Chowdhury et al. 2015. *Metabolites* 5, 4 (2015), 536–570.
[7] Kalyanmoy Deb et al. 2002. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.
[8] Patrick Kemmeren et al. 2014. *Cell* 157, 3 (2014), 740–752.
[9] Max Kuhn et al. 2018. https://CRAN.R-project.org/package=caret.
[10] Jiazhang Lian et al. 2018. *Metabolic Engineering* 50 (2018), 85–108.
[11] Jeffrey D Orth et al. 2010. *Nature biotechnology* 28, 3 (2010), 245.
[12] Noah Simon et al. 2013. *Journal of Computational and Graphical Statistics* 22, 2 (2013), 231–245.

Towards Detection of Engineering in Metagenomic Sequencing Data for Yeast and Other Fungi

Sancar Adali[1], Aaron Adler[1], Joel S. Bader[2], John Grothendieck[1], Thomas Mitchell[1], Anton Persikov[3], Jonathan Prokos[2], Richard Schwartz[1], Mona Singh[3], Allison Taggart[1], Benjamin Toll[1], Stavros Tsakalidis[1], Daniel Wyschogrod[1], Fusun Yaman[1], Eric Young[4], and Nicholas Roehner[1]

[1]Raytheon BBN Technologies, [2]Johns Hopkins University, [3]Princeton University, [4]Worcester Polytechnic Institute

nicholas.roehner@raytheon.com

1 INTRODUCTION

While there are many potential benefits of synthetic biology, including new applications in medicine, energy, and agriculture, these benefits do not come without tangible risks due to the possibility of accidental release or deliberate misuse of engineered organisms. Fungal pathogens are already engineered for beneficial lethal ends, such as biopesticides to protect crops [3], but could have unintended consequences with respect to the environment. In terms of deliberate misuse, engineering enhancement of existing pathogens to develop anti-crop bioweapons is of particular concern. Natural fungi are among the most common causes of infections in crops [1], have been successfully militarized in the past, and could be made more deadly using synthetic biology.

Hindering any effective response to the above is the lack of technology designed specifically to alert us when a novel organism (engineered or natural) is present in the environment. Detection of engineered organisms is challenging due to a combination of factors: the large space of existing but as of yet unknown natural DNA, the complexity of the environmental background in terms of mixed species populations, and the potential subtlety of some signatures of engineering. Possible solutions include training machine learning systems to recognize common patterns of engineering, employing single-cell sequencing to demultiplex metagenomic samples [2], and integrating the results of multiple approaches that capture a wide variety of signatures. We are developing a system called GUARDIAN (Guard for Uncovering Accidental Release, Detecting Intentional Alterations, and Nefariousness) that combines wetware and software solutions like these for the purpose of detecting even subtle signatures of engineering in metagenomic samples.

2 THE GUARDIAN SOFTWARE SYSTEM

The GUARDIAN software system can be divided into five distinct subsystems that implement approaches in two classes with different strengths and weaknesses: knowledge-free and knowledge-rich. Our knowledge-free approaches extract and classify signatures of engineering using methods from cybersecurity, natural language processing, and machine learning. Alternatively, our knowledge-rich approaches use methods from bioinformatics and genomics to accomplish these tasks. Figure 1 shows how these subsystems are connected and share information to determine whether a given sequencing data set was collected from a sample containing engineered organisms. Currently, GUARDIAN is targeted towards the detection of engineering in industrially relevant yeasts and other fungi such as *S. cerevisiae*, *Y. lipolytica*, and *P. pastoris*, but could be adapted to other taxa given appropriate training data. Here we focus on our knowledge-free approaches, which currently drive GUARDIAN's overall performance, and omit the knowledge-rich approaches, which currently supply additional details about detected engineering.

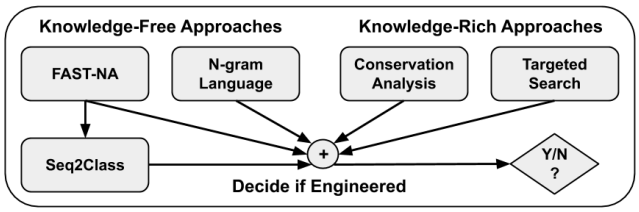


Figure 1: GUARDIAN system architecture. Answers are combined via weighted and unweighted voting schemes.

FAST-NA, a tool originally developed for detecting malware in network traffic, has since been adapted to screen DNA synthesis orders for dangerous sequences and screen natural DNA sequences for signatures of engineering. FAST-NA uses a probabilistic data structure known as a Bloom filter to identify regions within tested contigs that resemble short subsequences of the natural genomes in its training set. The remaining unidentified regions are then flagged as regions of interest (ROIs). To produce a yes/no answer for whether a sequencing data set contains engineering, we compute statistics over all ROIs detected with FAST-NA and compare them to those computed for known non-engineered genomes. For example, max ROI length can be used to detect engineering features such as large insertions.

Seq2Class is an approach that uses deep learning models trained on positive examples of engineering, such as plasmid vectors or insertions of refactored gene clusters, to compute

engineering scores for a sliding window over tested contigs. We then compute the fraction of these scores that fall above an arbitrarily chosen threshold and call this fraction the *DF* statistic. The final decision on engineered versus not engineered is made by comparing the *DF* statistic for a sequencing data set to the largest *DF* statistics computed for known non-engineered genomes. In order to increase the sensitivity of these models, we provide the ROIs detected with FAST-NA as input, thereby focusing on only those sequences that are likely candidates for containing signatures of engineering.

N-gram language models can be used to compute the likelihood of the N^{th} base pair given the previous sequence of $N-1$ base pairs. We have trained these models on examples of non-engineered genomes and used them to compute the sequence entropy of tested contigs. The higher the entropy computed for a sequence, the less closely it resembles the non-engineered genomes in the training set. We use a sliding window to compute ROIs containing base pairs belonging to high entropy sequences, and we compute statistics over these ROIs similar to those produced by FAST-NA. We then compare these ROI statistics to those for non-engineered genomes to make the decision of engineered versus not engineered.

3 RESULTS AND DISCUSSION

We tested the GUARDIAN software system on sequencing data sets for 28 samples of *S. cerevisiae* (25 engineered, 3 non-engineered) and 22 samples of *Y. lipolytica* (10 engineered, 12 non-engineered). Figure 2 shows the accuracy for each knowledge-free subsystem in GUARDIAN as well as the accuracy for the overall integrated system. Both FAST-NA and N-Gram had significantly greater accuracy when applied to *S. cerevisiae* compared to *Y. lipolytica*. One likely explanation for this observation is the smaller volume of negative training data available for *Y. lipolytica* versus *S. cerevisiae* (10s of genomes versus 1,000 [4]), leading to a greater number of false positives and decreased accuracy. Overall, the integrated system had 100% accuracy for *S. cerevisiae* samples, 95% for *Y. lipolytica* samples, and 98% for all samples.

The Seq2Class subsystem, on the other hand, had greater accuracy when applied to *Y. lipolytica*. To better understand this, we examined the sensitivity (true positive rate) and specificity (true negative rate) for each subsystem (Figure 3). Besides having slightly greater sensitivity than either FAST-NA or N-Gram for *Y. lipolytica*, Seq2Class had much higher specificity, indicating far fewer false positives. Given that Seq2Class takes the output of FAST-NA as input, this suggests that applying deep learning models trained on positive examples of engineering to regions obtained by filtering with negative examples is a promising means of increasing specificity, without necessitating the collection of additional

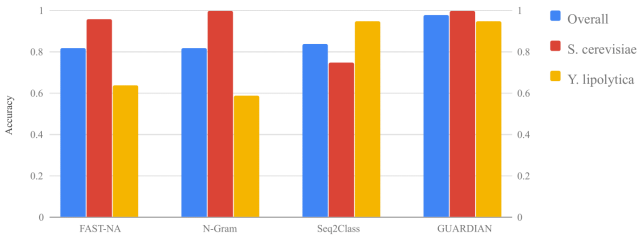


Figure 2: Accuracy of overall GUARDIAN system and individual subsystems when applied to *S. cerevisiae* and *Y. lipolytica*.

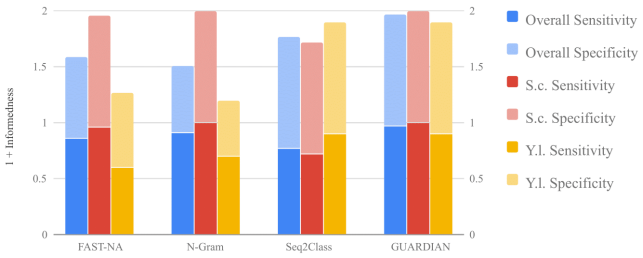


Figure 3: Informedness + 1 of GUARDIAN system when applied to *S. cerevisiae* and *Y. lipolytica*. A value of 1 indicates performance no better than random.

negative training data. This is especially attractive as we can generate synthetic positive training data *in silico*.

To date, we have tested GUARDIAN on samples of moderate complexity that contain single species of fungi or, in the case of two samples, fungi in the presence of non-engineered, non-fungal DNA. In the future, we will test GUARDIAN on more complex metagenomic samples, some simulated.

ACKNOWLEDGMENTS

We thank Pacific Northwest National Laboratory and Lawrence Berkeley National Laboratory for preparing the sequencing data with which we tested GUARDIAN. This work was supported by the IARPA Finding Engineering-Linked Indicators (FELIX) award HR0011-15-C-0084. This document does not contain technology or technical data controlled under either U.S. International Traffic in Arms Regulation or U.S. Export Administration Regulations.

REFERENCES

[1] Fisher, M. C., et al. Emerging fungal threats to animal, plant and ecosystem health. *Nature 484* (2012), 186–194.

[2] Lan, F., et al. Single-cell genome sequencing at ultra-high-throughput with microfluidic droplet barcoding. *Nature Biotechnology 35* (2017), 640–646.

[3] Lovett, B., and Leger, R. J. S. Genetically engineering better fungal biopesticides. *Pest Management Science 74* (2017).

[4] Peter, J., et al. Genome evolution across 1,011 Saccharomyces cerevisiae isolates. *Nature 556* (2018), 339–344.

Biophysical Analysis for Implementing Neuro-inspired Computing in Living Cells

Ramez Daniel
Luna Rizik
ramizda@bm.technion.ac.il
luna.rizik@gmail.com
Technion - Israel Institute of Technology
Haifa, Israel

Ximing Li
Technion - Israel Tnstitute of Technology
Haifa, Israel
ximing.li@campus.technion.ac.il

Raghd Abu Sinni
Technion - Israel Tnstitute of Technology
Haifa, Israel
sraghd@campus.technion.ac.il

1 INTRODUCTION

The field of synthetic biology builds genetic circuits in living cells by implementing design principles of engineering, in order to construct new functionalities for biotechnological and biomedical applications [1]. The dominant computing paradigm in synthetic biology is digital, involving computation with two discrete (Boolean) numbers for inputs and outputs (OFF and ON) such as switches, counters, logic gates, memories, and edge detectors [5]. significant progress has been made in creating orthogonal synthetic biological parts (e.g., TFs, promoters, recombinase and RNA devices) that are assembled to construct complex logic functions [4]. However, scaling the complexity of digital circuits is severely limited because it requires a large number of parts, and high levels of protein expression, while placing substantial metabolic and toxicity burdens on the host cells. In efforts to scale up the computational complexity of genetic circuits, an analog computing was suggested and implemented in living cells [3]. This analog paradigm computes with a continuous set of numbers, and scales significantly more efficiently than its digital counterpart. However, analogous to electronics, design issues such as noise, reliability, and loading effects have pushed circuits to their limits in scaling of computation. By contrast to synthetic computation in living cells, natural biological systems consist of imprecise units that collectively interact with each other to reliably perform computationally intensive tasks (Fig. 1).

2 PERCEPTUAL COMPUTING MODELS IN LIVING CELLS

The basic computational element in ANNs is the perceptron [7] (Fig. 2a), which performs the following three operations. (1) Signal weighting: multiplication of each analog input (x_i) by a scalar (w_i) that represents the synaptic weight. (2) Signal aggregation: summing of all the inputs in an analog fashion. (3) Thresholding: non-linear activation of the summed analog signal. The perceptron's output is given by:

$$y=\sum_{i=1}^{N} w_i \cdot x_i \quad (1.1) \qquad z=\frac{e^y}{1+e^y} \quad (1.2)$$

In Eq.1.1 the parameter b is used for biasing, and N is number of the inputs. The perceptron is a model that captures the computational power of neurons (parallelism and adaptivity [7]). Fig. 1 has shown that genetic and neural architectures share some computational features e.g., decision is made based on collective interaction of analog bio-molecules. Thus, our next step is to describe the basic computational structure of genetic processing units using a transformative version of the perceptron. In biological systems, the interactions between proteins and DNA that control promoter activity

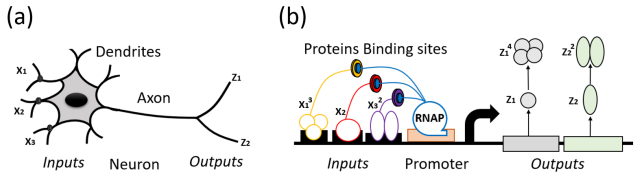


Figure 1: (a) neural network [7] (b) gene network [6].

can be realized as sigmoid activation functions operating in the logarithmic domain, and can be described by Michaelis-Menten kinetics at the steady state [3]. Therefore, analogously, scalar multiplication and summation in the perceptron are converted, respectively, to power-law and multiplication:

$$Y=\prod_{i=1}^{N} B \cdot \left(\frac{x_i}{K_i}\right)^{n_i} \quad (2.1) \qquad z=\frac{e^{m \cdot ln(Y/k_d)}+\beta}{1+e^{m \cdot ln(Y/k_d)}+\beta} \quad (2.2)$$

where Y is a multi-protein complex, X_i is the concentration of transcription factor (TF), K_i is a normalization constant (or input dynamic range of X_i), n_i is known as the Hill coefficient, that describes cooperativity, and $B = \alpha \cdot \tau$ has a unit of concentration (α protein production rate and τ protein half time). Eq. 2.2 is known as a Hill-function, where K_d is a dissociation constant of binding reaction, β is the basal level of the promoter, m is number of binding sites in the promoter, and Z is the promoter activity. We coined the new model (Eq. 2, Fig. 2b) "perceptgene", and it can be viewed as a perceptron that operates in the logarithmic domain. We compared

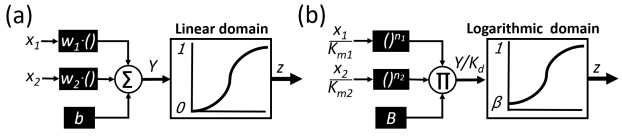


Figure 2: (a) perceptron, widely used in ANNs, (b) perceptgene, new developed that is suitable for gene circuits

three perceptual computing models, (1) perceptron (Eq. 1.1-1.2), (2) perceptgene (Eq. 2.1-2.2) and (3) a hybrid model, a perceptron with an activation function of perceptgene (Eq. 1.1, 2.2). Our simulation results show that the perceptron and perceptgene have a similar behavior; both unambiguously separated the analog pattern into two discrete levels (Fig. 3b-3d). By contrast, the hybrid model failed to act as a binary classifier. The hybrid model has a simple genetic

structure but it requires biological parameters that are challenging to achieve.

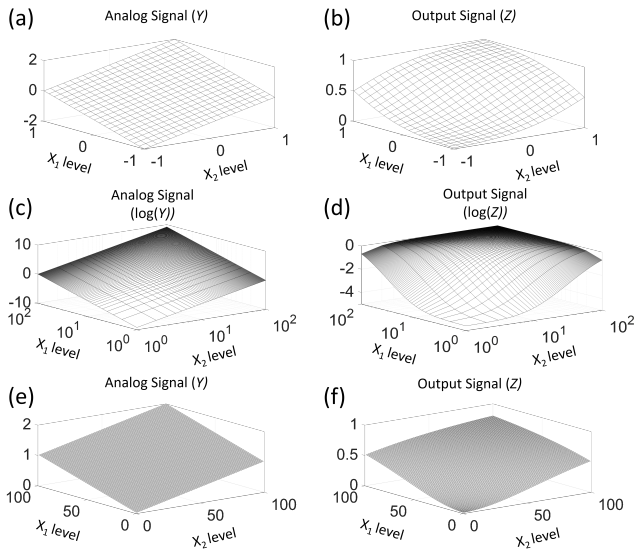


Figure 3: Simulation results of perceptual computing models, with the analog signal (Y) and the corresponding output signals (Z): (I) perceptron: Eq. 1.1-1.2, Fig (a)-(b). (II) perceptgene: Eq. 2.1-2.2, Fig (c)-(d). (III) hybrid model Eq. 1.1-2.2, Fig (e)-(f).

3 NEURO-INSPIRED COMPUTATION IN LIVING CELLS

The first step toward the implementation of neuro-inspired genetic circuits is to build weighted analog pattern that consists of a power-law and multiplication function. Fig. 4a shows a one possible design, and it includes auto-negative feedback (ANF) loops [3] and a combinatorial promoter [2]. The analog signal (Y) is represented by the activity of the combinatorial promoter. The circuit activity at steady state:

$$R_i \cdot (1 + (\frac{X_i}{K_{mi}})^{h_i}) = \frac{\alpha \cdot \tau}{1 + (\frac{R_i}{K_d i})^{n_i}} \tag{3.1}$$

$$Y = \frac{\alpha \cdot \tau}{1 + (\frac{R_1}{K_d 1})^{n_1} + (\frac{R_2}{K_d 2})^{n_2} + (\frac{R_3}{K_d 3})^{n_3} + (\frac{R_4}{K_d 4})^{n_4}} \tag{3.2}$$

where i=1,2, h_i is the Hill-coefficient of binding inducer X_i to repressor R_i, K_{di} is the dissociation constant of binding the repressor R_i to promoter, K_{mi} is the dissociation constant of binding the inducer X_i to repressor R_i. Here, we assumed that there is no crosstalk between the repressors. The simulation results of Eq. 3 are shown in Fig. 4b and exhibit a power-law and multiplication function. To implement the perceptgene, we connected the power-law and multiplication circuit with an activator that has a non-linear behavior (Fig. 4a). The perceptgene output (Z) is represented by the activator promoter, and is given by Eq.2.2 and Eq.3. Fig. 4c shows that we can implement a genetically encoded perceptron operates in the logarithmic domain. To control the design parameters of our circuits, we developed a perceptgene-based rule, in analogy to the supervised learning algorithms of ANNs. The new learning algorithm is based on: (1) computation of the perceptgene depends on the Hill-coefficients and by adjusting their values, we can obtain a wide range of possible target output for specific inputs, (2) the training rule minimizes the output error, using a gradient descent in a log-linear domain:

$$< E >= \frac{1}{N} \sum i = 1N \frac{(log(z_D i) - log(z_i))^2}{2} \tag{3.3}$$

Eq. 4 calculates the average error between the desired data (Z_{Di}) and the actual perceptgene output (Z), N is the number of samples. By using a chain-rule, in addition to gradient-descent (Eq. 4), we calculated the update changes to hi and n_i, and trained the circuit of Fig. 4a to exhibit AND gate (<E>=3%) and OR gate (<E>=0.3%) as shown in Fig. 4d. In summary, the perceptgene is an abstract model that can be implemented in living cells to build neural gene networks. These networks can be trained by updating their Hill-coefficients acting as weights.

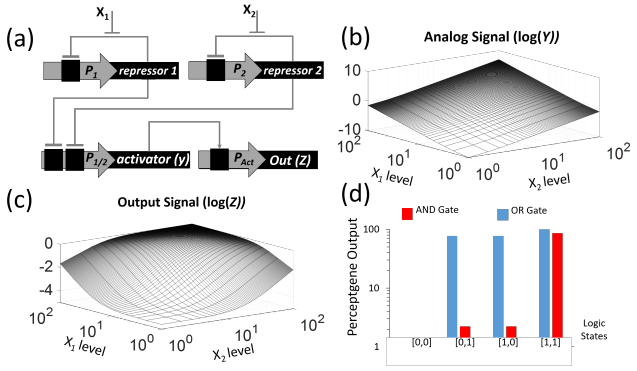


Figure 4: Figure 4: (a) Implementation of perceptgene in living cells consists of auto-negative feedback loops (ANF) and an activator. (b) Simulation result of ANF with parameters $h_i = 2, n_i = 1, m = 2, K_{di} = 10, K_{mi} = 1, \alpha \cdot \tau = 10, K_d = 3000$. (c) Simulation result of perceptgene. (d) Implementation of logic gates using perceptgene and gradient decent algorithm, with simulation parameters: $m = 2, K_{di} = 10, K_{mi} = 1, \alpha \cdot \tau = 10, K_d = 4000$. AND gate: $h_i = 1.5, n_i = 1$. OR gate: $h_i = 2, n_i = 2z$

REFERENCES

[1] Brophy, J. A., and Voigt, C. A. Principles of genetic circuit design. *Nature methods 11*, 5 (2014), 508.
[2] Cox, R. S., Surette, M. G., and Elowitz, M. B. Programming gene expression with combinatorial promoters. *Molecular systems biology 3*, 1 (2007), 145.
[3] Daniel, R., Rubens, J. R., Sarpeshkar, R., and Lu, T. K. Synthetic analog computation in living cells. *Nature 497*, 7451 (2013), 619.
[4] Green, A. A., Kim, J., Ma, D., Silver, P. A., Collins, J. J., and Yin, P. Complex cellular logic computation using ribocomputing devices. *Nature 548*, 7665 (2017), 117.
[5] Lamport, L. *Synthetic Gene Networks. Encyclopedia of Molecular Cell Biology and Molecular Medicine.* Addison-wesley, 1994.
[6] Ptashne, M. How eukaryotic transcriptional activators work. *Nature 335*, 6192 (1988), 683.
[7] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review 65*, 6 (1958), 386.

Analyzing Genetic Circuits for Hazards and Glitches

Pedro Fontanarrosa
pfontanarrosa@gmail.com
University of Utah
Salt Lake City, UT, USA

Hamid Hosseini, Amin Borujeni, Yuval Dorfan, Chris Voigt
Massachusetts Institute of Technology
Boston, MA, USA

Chris Myers
myers@ece.utah.edu
University of Utah
Salt Lake City, Utah, USA

1 INTRODUCTION

A *hazard* is the possibility of an unwanted or unexpected variation of the output of a combinational logic network before it reaches steady-state. A *glitch* is the actual observance of such a problem. These terms are used mostly for electronic circuits, though glitches have been observed for *genetic regulatory networks* (GRNs) as well. A glitch is a transient behavior that corrects itself as the system reaches a steady-state. Nonetheless, this glitching behavior can have drastic consequences if this transient output of the GRN causes an irreversible change in the cell such as a cascade of responses within or with other cells, or if it induces apoptosis. Therefore, avoiding glitching behavior can be crucial for safe operation of a genetic circuit.

To better understand glitching behavior in genetic circuits, this paper utilizes a version of our dynamic model generator [2] that automatically generates a mathematical model composed of a set of *ordinary differential equations* (ODEs) that are parameterized using characterization data found in a Cello genetic gate library [4]. Simulation of a dynamic model allows for the prediction of glitches that cannot be observed with steady-state analysis.

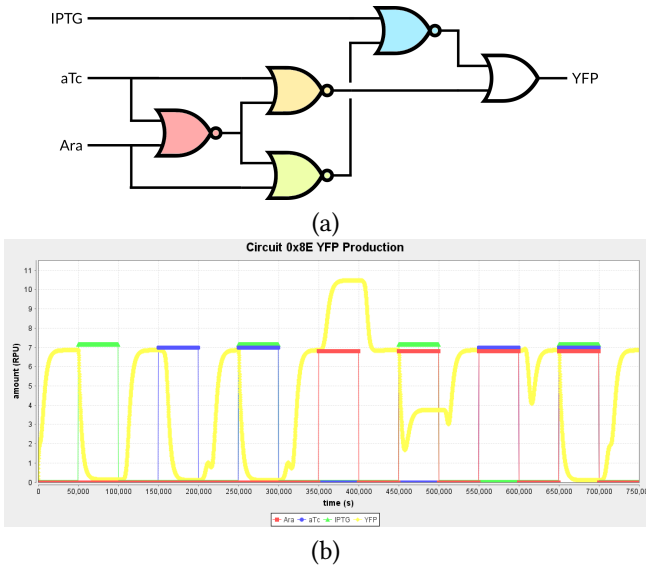


Figure 1: (a) Circuit diagram for circuit 0x8E from the Cello paper [4]. (b) YFP production (in *Relative Promoter Units*) over time (in *seconds*) for circuit 0x8E for each combination of input molecules (IPTG, aTc, Ara). Simulation obtained using the automatic model generated in iBioSim [2].

2 CIRCUIT ANALYSIS

To demonstrate our analysis procedure, this paper analyzes circuit 0x8E (see Figure 1(a)) from the Cello paper [4], since this paper gave experimental time course data that included glitching behavior. The behavior of this circuit that is predicted using our automatically generated model is shown on Figure 1(b), in which *Yellow Fluorescent Protein* (YFP) production in *Relative Promoter Units* (RPU) [1] is shown over time for each possible input value. The simulation shows glitches on YFP production in several places including the condition observed experimentally (namely when Ara and IPTG are provided simultaneously).

To better understand the cause of this glitch, let us consider the Karnaugh map for this circuit shown in Table 1. The values in the Karnaugh map indicate the steady-state YFP production (1: *High*, 0: *Low*) for different combinations of input molecules. Let us first consider when the environment changes from no input molecules to where IPTG and aTc are *high*. In the Karnaugh map, this moves the circuit from (Ara, IPTG, aTc) = (0, 0, 0) to (0, 1, 1). Since there are two input changes, there are two different paths from the initial-state to the end-state, depending on which input change the circuit *"senses"* first. If the circuit senses the aTc change first, the circuit momentarily passes through (0, 0, 1), while if the circuit senses IPTG change first, the circuit momentarily passes through (0, 1, 0). In both of these states and the final state, the circuit evaluates to *low*, so the circuit makes a monotonic change from 1 to 0, and there is no possibility of a glitch. Now, let us consider a transition from no input molecules (0, 0, 0) to where Ara and IPTG are *high* (1, 1, 0). If the circuit senses first the Ara input molecule, it passes through (1, 0, 0) before reaching (1, 1, 0), and it evaluates to *high* in all these states. However, if the circuit senses IPTG before it senses Ara, it will momentarily pass through (0, 1, 0), which evaluates to a *low* output, before reaching the end-state where the output is *high*, thus producing the glitch that is observed both in the simulation and the experimental results. This potential for a glitch is known in the asynchronous logic community as a *function hazard* [3]. The existence of a function hazard means that regardless of how the circuit is designed, the possibility of a glitch remains. In other words, a function

hazard is a property of the *function* and not of the circuit implementation. However, it is not always the case that the existence of a function hazard means that a glitch occurs. The transition from no inputs (0, 0, 0) to Ara and aTc *high* (1, 0, 1) also implies two input changes, thus two different paths from the initial-state to the end-state. The order that these input changes are sensed affects the output of the circuit; therefore a function hazard exists. Yet, the glitching behavior is not observed in the simulation shown in Figure 1(b).

Table 1: Karnaugh Map for Circuit 0x8E

Ara \ IPTG aTc	0 0	0 1	1 1	1 0
0	1	0	0	0
1	1	1	0	1

Figure 2(a) (yellow line) shows the simulation of all the 2-input change function hazards for this circuit. It can be observed that multiple glitches occur. Glitches manifest when multiple inputs pass through multiple paths with different delays. Therefore, a possible solution to remove the glitch could be to add more delay to the shorter path as shown Figure 2(b). Simulating this modified circuit shows that adding the redundant logic avoids some glitches but not all of them, and the response is slower (Figure 2 (a), red line). The only way to avoid function hazards is to restrict the allowed input changes to the system. As an example of this, limiting the input changes to only *single* input changes produces a very smooth and glitch-free function, as shown by the simulation in Figure 2(c).

3 DISCUSSION

This paper presents a method of analysis for hazards and glitches in genetic circuits that leverages dynamic model generation. It illustrates this procedure using the Cello 0x8E circuit that exhibited glitching behavior in experimental results. We identified the cause of the problem as a function hazard in the circuit caused by multiple input changes. A system with a function hazard always has the potential to glitch for the specified input change, and modifying the implementation can only change its likelihood of occurring. For example, we demonstrated how adding some extra redundant logic to add delay to the short path of the circuit may reduce the likelihood of some glitches while increasing it for others. The only solution to avoid function hazards completely is to restrict the allowed input changes to the system to include only single-input changes and a restricted set of multiple input changes. In the future, we plan to map the RPU units to actual molecule count to enable stochastic analysis for prediction of the probability of glitches.

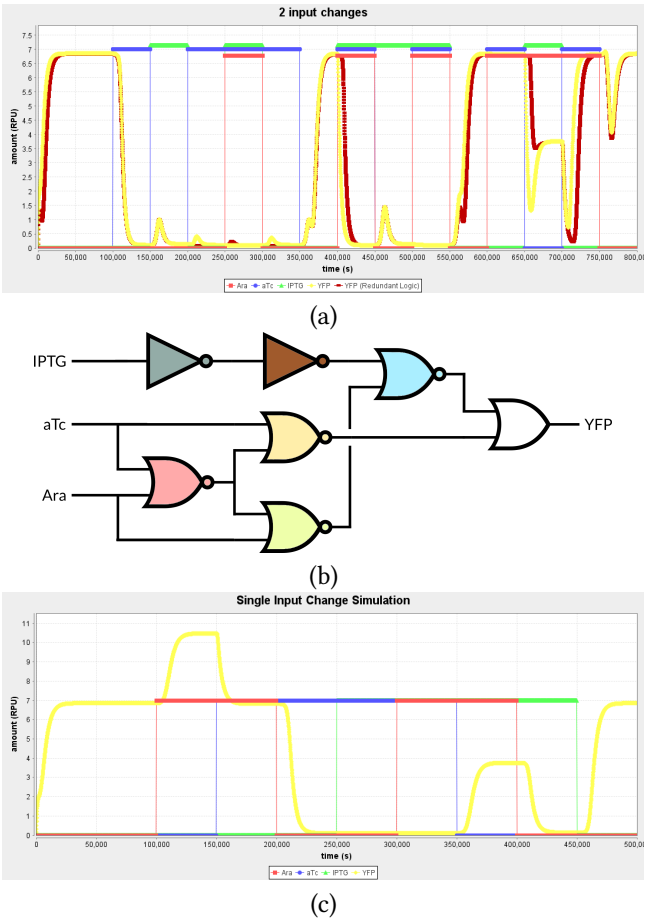


Figure 2: (a) Simulation for all 2-input change function hazards. (b) Circuit 0x8E with redundant logic. (c) Simulation of original circuit with single input changes.

Acknowledgements

We would like to thank Tim Jones, Dany Fu, and Douglas Densmore of Boston University for their feedback on this work. The authors of this work are supported by DARPA FA8750-17-C-0229.

REFERENCES

[1] Kelly, J. R., Rubin, A. J., Davis, J. H., Ajo-Franklin, C. M., Cumbers, J., Czar, M. J., de Mora, K., Glieberman, A. L., Monie, D. D., and Endy, D. Measuring the activity of biobrick promoters using an in vivo reference standard. *Journal of biological engineering 3*, 1 (2009), 4.

[2] Misirli, G., Nguyen, T., McLaughlin, J. A., Vaidyanathan, P., Jones, T. S., Densmore, D., Myers, C., and Wipat, A. A computational workflow for the automated generation of models of genetic designs. *ACS synthetic biology* (2018).

[3] Myers, C. J. *Asynchronous circuit design.* John Wiley & Sons, 2001.

[4] Nielsen, A. A., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. Genetic circuit design automation. *Science 352*, 6281 (2016), aac7341.

An automated model reduction tool to guide the design and analysis of synthetic biological circuits

Ayush Pandey*
Richard M. Murray†

1 INTRODUCTION

We present an automated model reduction algorithm that uses quasi-steady state approximation based reduction to minimize the error between the desired outputs. Additionally, the algorithm minimizes the sensitivity of the error with respect to parameters to ensure robust performance of the reduced model in the presence of parametric uncertainties. We develop the theory for this model reduction algorithm and present the implementation of the algorithm that can be used to perform model reduction of given SBML models. To demonstrate the utility of this algorithm, we consider a synthetic biological circuit to control the population density and composition of a consortium consisting of two different cell strains. For this example, we show the application of our algorithm that minimizes the sensitivity of the error along with the error itself to return a final reduced model that has robust performance to parametric uncertainties.

2 PROBLEM FORMULATION

For a state vector $x \in \mathbb{R}^n$, a vector consisting of all model parameters $\Theta \in \mathbb{R}^p$ and the vector of output measurements $y \in \mathbb{R}^k$, we define the nonlinear dynamic model with a nonlinear function vector f and a $k \times n$ matrix H,

$$\dot{x} = f(x, \Theta),\ y = Cx.$$

Our goal is to find a reduced model with a lower dimensional state vector $\hat{x} \in \mathbb{R}^{\hat{n}}$ so that $\hat{n} < n$,

$$\dot{\hat{x}} = \hat{f}(\hat{x}, \Theta),\ \hat{y} = \hat{C}\hat{x},$$

for initial conditions $x(0) = x_0,\ \hat{x}(0) = \hat{x}_0$ and $\hat{y} \in \mathbb{R}^k$ that minimizes the error $e \triangleq y - \hat{y}$ and the sensitivity of the error $S_e \triangleq \frac{\partial e}{\partial \theta}$ for all $\theta \in \Theta$. To formulate the sensitivity minimization problem, we define sensitivity coefficients of states with respect to a parameter θ as $s_i = \frac{\partial x_i}{\partial \theta}$. We have from [1] that $\dot{S} = JS + Z$ where J is the Jacobian of the system, S is the vector of the sensitivity coefficients and Z is the vector of the partial derivatives of the dynamics, $\frac{\partial f}{\partial \theta}$. Similarly, we have $\dot{\hat{S}} = \hat{J}\hat{S} + \hat{Z}$ for any reduced model. In our approach, we collapse some of the states' dynamics into algebraic relations, hence, retaining the structure and the meaning of the states of the original model. This process is reminiscent of singular perturbation based model reduction algorithms [2]. Let x_c denote the vector of collapsed states. Then, for a permutation matrix T that consists of only 0s and 1s with the condition that there can only be one non-zero element in a row and a column, we have,

$$x = T \begin{bmatrix} \hat{x} \\ x_c \end{bmatrix} = \begin{bmatrix} T_1 & T_2 \end{bmatrix} \begin{bmatrix} \hat{x} \\ x_c \end{bmatrix},\ T_1 \in \mathbb{R}^{n \times \hat{n}},\ T_2 \in \mathbb{R}^{n \times (n-\hat{n})}.$$

3 MAIN RESULT

We give a method to upper bound the norm of the sensitivity of the error that can be computationally more efficient than the brute force sensitivity analysis of all possible reduced order models to compute the sensitivity of error.

Theorem 1. *Suppose that there is a matrix $P = P^T \geq 0$ that solves the continuous-time Lyapunov equation for $\bar{J} = \mathrm{diag}(J, \hat{J})$ i.e., $\bar{J}^T P \bar{J} + P\bar{J} = -\bar{C}^T \bar{C}$, where $C = \begin{bmatrix} C & -\hat{C} \end{bmatrix}$, then the norm of the sensitivity of the error for the nonlinear model reduction can be upper bounded for some $N > 0$,*

$$\|S_e\|_2^2 \leq \lambda_{\max}(P) + 2N \max_t \left\| \begin{bmatrix} \frac{\partial f}{\partial \theta} \\ \frac{\partial \hat{f}}{\partial \theta} \end{bmatrix} Q_s \begin{bmatrix} \frac{\partial \hat{x}}{\partial \theta} \\ \frac{\partial x_c}{\partial \theta} \end{bmatrix} \right\|_2, \quad (1)$$

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix},\ Q_s = \begin{bmatrix} P_{11}T_1 + P_{12} & P_{11}T_2 \\ P_{21}T_1 + P_{22} & P_{21}T_2 \end{bmatrix}.$$

Proof. See [5]. □

4 ALGORITHM

We develop a brute force algorithm [4] based on the results presented in the previous section to compute the sensitivity of the error efficiently for all reduced models. The algorithm iterates over all possible reduced models by collapsing a subset of state dynamics. It computes the error and its corresponding sensitivity bound to return a final reduced model that satisfies the user specified tolerance levels.

5 EXAMPLE

Population and composition control circuit

In this example, two coupled feedback controllers are designed using various genetic components that control the total population density of a bacterial consortium to a desired value and the fraction of the two cell types in the consortium.

*Department of Electrical Engineering at California Institute of Technology, Pasadena, CA, USA. Email - apandey@caltech.edu
†Department of Control and Dynamical Systems at California Institute of Technology, Pasadena, CA, USA. Email - murray@cds.caltech.edu

See [3] for the details of the implementation of this circuit. The full ODE model is given in equation 2.

$$
\begin{aligned}
\frac{dN_1}{dt} &= \beta_{R_1}\left(l_{R_1} + \frac{R_1^2}{K_{R_1}+R_1^2}\right) - k_b A_1 N_1 - d_T N_1 \\
\frac{dA_1}{dt} &= K_r\beta_{R_2}\left(l_{R_2} + \frac{R_2^2}{K_{R_2}+R_2^2}\right) - k_b A_1 N_1 - d_T A_1 \\
\frac{dR_1}{dt} &= \beta_{tac}\left(l_{tac} + \frac{I^2}{K_{tac}+I^2}\right) L_1 - d_S R_1 \\
\frac{dR_2}{dt} &= \beta_{sal}\left(l_{sal} + \frac{L^2}{K_{sal}+L^2}\right) L_2 - d_S R_2 \\
\frac{dN_2}{dt} &= \beta_{R_2}\left(l_{R_2} + \frac{R_2^2}{K_{R_2}+R_2^2}\right) - k_b A_2 N_2 - d_T N_2 \\
\frac{dA_2}{dt} &= K_r\beta_{R_1}\left(l_{R_1} + \frac{R_1^2}{K_{R_1}+R_1^2}\right) - k_b A_2 N_2 - d_T A_2 \\
\frac{dL_1}{dt} &= k_C\left(1 - \frac{L_1+L_2}{C_{max}}\right) L_1 - d_c L_1 \frac{N_1}{K_{tox}+N_1} - dL_1 \\
\frac{dL_2}{dt} &= k_C\left(1 - \frac{L_1+L_2}{C_{max}}\right) L_2 - d_c L_2 \frac{N_2}{K_{tox}+N_2} - dL_2.
\end{aligned}
\tag{2}
$$

Here, the state vector x consists of the toxins (N_i), signals (R_i) and the anti-toxins (A_i) species corresponding to each cell type (L_1 and L_2).
We have, $x = \begin{bmatrix} N_1 & A_1 & R_1 & R_2 & N_2 & A_2 & L_1 & L_2 \end{bmatrix}^T$.

$$
\begin{aligned}
\hat{f}_1 &= \left(l_{R_1} + \frac{\beta_{R_1} x_7^2}{x_7^2 + K_{I0}}\right) - d_T x_1 \\
&\quad - \frac{\beta_{R_2} k_b x_1 \left(K_{a0} l_{R_2} + x_8^2\right)}{k_b x_1 x_8^2 + K_{a0} k_b x_1 + d_T x_8^2 + K_{a0} d_T}, \\
\hat{f}_2 &= \left(l_{R_2} + \frac{\beta_{R_2} x_8^2}{x_7^2 + K_{a0}}\right) - d_T x_5 \\
&\quad - \frac{\beta_{R_1} k_b x_5 \left(K_{I0} l_{R_1} + x_7^2\right)}{k_b x_5 x_7^2 + K_{I0} k_b x_5 + d_T x_7^2 + K_{I0} d_T}, \\
\hat{f}_3 &= k_c\left(1 - \frac{x_7+x_8}{C_{max}}\right) x_7 - \frac{d_c x_1 x_7}{x_1 + K_{tox}} - d x_7, \\
\hat{f}_4 &= k_c\left(1 - \frac{x_7+x_8}{C_{max}}\right) x_8 - \frac{d_c x_5 x_8}{x_5 + K_{tox}} - d x_8.
\end{aligned}
\tag{3}
$$

Using our model reduction algorithm, we obtain final reduced order model as shown in equation 3. This model has the best robust performance among the multiple possible reduced models as shown in Figure 1 and Figure 2.

The project or effort depicted was or is sponsored by the Defense Advanced Research Projects Agency (Agreement HR0011-17-2-0008). The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

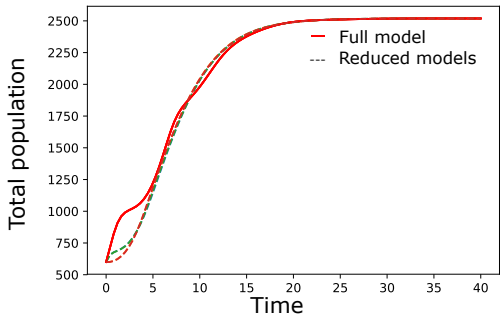


Figure 1: Comparison of the total population in the full model and the reduced models obtained when considering only the error metric. It is clear that all of the four reduced models have similar performance (note that the curves for two of the models are stacked on top of others) and so the choice of the reduced model is not clear from this metric alone. However, on using our algorithm that minimizes the sensitivity of the error (shown in Figure 2) as well, we get a final reduced model given in equation 3.

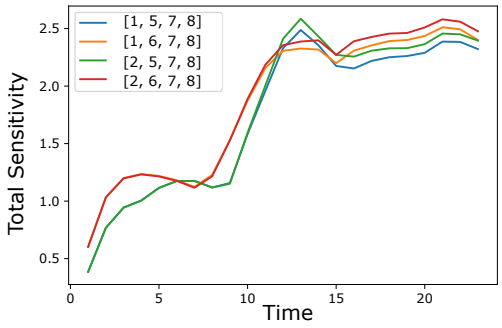


Figure 2: The total sensitivity of the population of L_2, with time for all four reduced models. The legend indicates the state indices that form the reduced model state vector $\hat{x}$. Observe that the reduced model with $\hat{x} = \begin{bmatrix} x_1 & x_5 & x_7 & x_8 \end{bmatrix}^T$ (given in equation 3) has the lowest total sensitivity.

REFERENCES

[1] Dickinson, R. P., and Gelinas, R. J. Sensitivity analysis of ordinary differential equation systems - a direct method. *Journal of computational physics 21*, 2 (1976), 123–143.

[2] Kokotovic, P. V., O'Malley Jr, R. E., and Sannuti, P. Singular perturbations and order reduction in control theory - an overview. *Automatica 12*, 2 (1976), 123–132.

[3] McCardell, R. D., Pandey, A., and Murray, R. M. Control of density and composition in an engineered two-member bacterial community. *bioRxiv* (2019).

[4] Pandey, A. Python based tool for automated model reduction of sbml models. Available on Github. Accessed 13 Jun 2019.

[5] Pandey, A., and Murray, R. M. An automated model reduction tool to guide the design and analysis of synthetic biological circuits. *bioRxiv* (2019).

Estimating Biologically Relevant Network Structures from Time-series Data

Zoltan A. Tuza
z.tuza@imperial.ac.uk
Department of Bioengineering,
Imperial College London
London, UK

Guy-Bart Stan
g.stan@imperial.ac.uk
Department of Bioengineering,
Imperial College London
London, UK

High-throughput data acquisition in systems and synthetic biology leads to an abundance of data that need to be processed and aggregated into biologically relevant dynamical models. Building such models based on this wealth of data is of paramount importance to gain insight into an existing biological process, or to understand and optimize designs of synthetic biology constructs. However, building models manually for each data set is inconvenient and can become impractical for highly complex biological systems.

Looking at the spectrum of model-building processes: on one end, we have first-principle-based modeling, which has been a successful method to build models in many areas of science and engineering; however, this method for model construction might not scale well with the copious amount of available data. On the other end of the spectrum, we have machine learning methods, which can build complex input-output models using a vast amount of data; however, the resulting interconnections in the model are completely detached from first principles and therefore such models are rarely human interpretable. A middle ground approach is gaining traction in the literature where a dictionary of possible functions—assembled by domain experts—is presented to an algorithm along with the measured data. Then, based on these inputs, the algorithm builds the most suitable (e.g. parsimonious) combination of dictionary functions as a dynamical model [1, 3, 4, 6].

In this work, we combine this dictionary function-based approach with Chemical Reaction Network Theory (CRNT) to automatically generate dynamical models, as well as all their possible reaction graphs. Figure 2. summarizes the main steps of our approach.

For the model building part, we use Sparse Bayesian Learning (SBL), which offers methods to learn the sparsest set of dictionary functions necessary to capture the dynamics of a system, and at the same time automatically builds a nonlinear Ordinary Differential Equation (ODE) model.

Sparse Bayesian Learning was first developed in the signal processing community and has solid theoretical foundations such as proof of convergence and guarantees for sparse recovery, under certain conditions [8]. These foundations make it an appealing candidate for an automatic model reconstruction method for biological systems. Figure 1. shows the main components of the SBL algorithm.

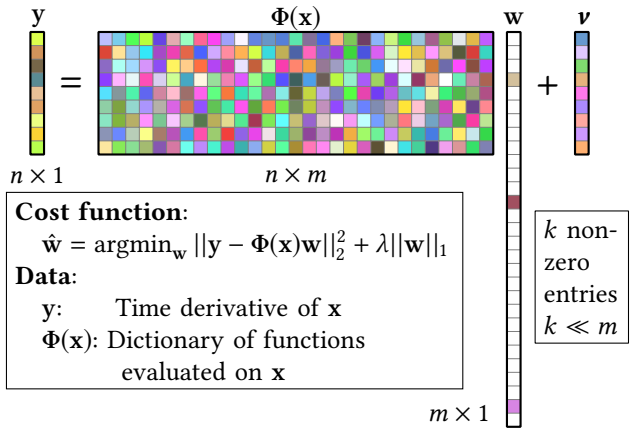


Figure 1: Sparse Bayesian Learning can estimate a vector of weights, which contains only a few nonzero entries by solving an underdetermined inverse problem. In our implementation, the vector y contains the derivative of the time-series data vector x. We also assemble a dictionary of possible model right-hand side terms in $\Phi(x)$, e.g. x_1x_2 or $\frac{x_1^n}{K+x_1^n}$. Such a dictionary comes from domain knowledge and it is evaluated on the time-series data. The vector v represents the measurement noise. The non-zero weights, which are computed by a series of convex optimization problems, select a few elements from the dictionary, and as a result a sparse model of the underlying dynamical process can be obtained.

At the core of the SBL algorithm, we have an iterative reweighted L_1 norm minimization, which associates nonzero weights to only a few dictionary functions. It should be emphasized that compared to common 'simulate-and-compare' approaches used for parameter (and model) estimation, this algorithm finds the parameters as well as the structure of the model after only a few dozens of iterations and does not require any resource intensive model simulation.

If the dynamical model describes a biochemical process, one might want to construct the reaction graph representation as well. It is usually assumed that there is a one-to-one

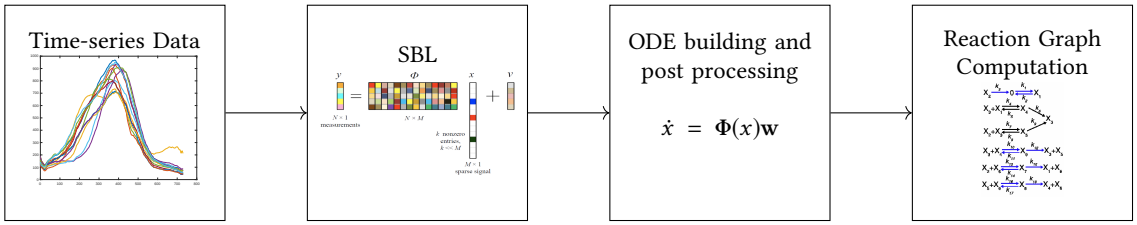


Figure 2: Workflow. First, a set of time-series data is collected from biological experiments. Given the processed data and the set of dictionary functions, the SBL algorithm estimates the weights of the dictionary functions. Then, a dynamical model is built from a sparse linear combination of dictionary functions. Finally, all graph structures that yield this same dynamical model can be computed.

correspondence between the dynamical model and the underlying network structure. However, this is only true if one assembles the differential equations from the network structure. Conversely, multiple non-equivalent graph structures can exhibit the same dynamics. The existence of multiple network structures for a given dynamics has been investigated extensively in [2, 5, 9]. However, these results were established under the assumption of perfect measurements. An extension of these results to the uncertain case has been presented recently [10]. As we show in this work, this extension allows us to use a sparse model structure, built from noisy time-series data, and to compute the possible network structures (Figure 3.).

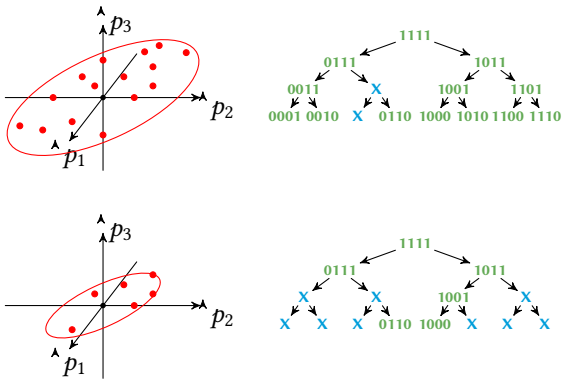


Figure 3: Left: the joint confidence region of the parameters are shown as red ellipses. Within the confidence region each red dot represents a valid CRN realization. Right: the possible network structures are shown as a binary tree. Each leaf (which corresponds a red dot on the left) is a binary string, where '1' means a reaction is present in a reaction graph—the reaction itself is identified by the index of the digit. The figure shows the relationship between the size of the confidence region and the number of possible network structures.

Additionally, the whole system identification process can be complemented with constraints on the parameters, for example, to enforce stability or non-negativity of the reconstructed models—thus offering relevant physical constraints over the possible network structures during the optimization process [7].

Using the above-described approach, the wealth of data can be translated into biologically relevant dynamical models as well as reaction graphs. On one hand, the set of possible graphs give an overview of the possible network connections and allows us to identify targets for further perturbation experiments (e.g. gene knockouts). On the other hand, the model at hand steers further data acquisition (e.g. provides inputs for optimal experimental design), thereby providing a vital step for closed-loop system identification.

REFERENCES

[1] Brunton, S. L., Proctor, J. L., and Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences 113*, 15 (mar 2016), 3932–3937.

[2] Johnston, M. D., Siegel, D., and Szederkényi, G. Dynamical equivalence and linear conjugacy of chemical reaction networks: new results and methods. *MATCH Commun. Math. Comput. Chem. 68* (2012), 443–468.

[3] Pan, W. *Bayesian Learning for Nonlinear System Identification.* PhD thesis, Imperial College London, 2017.

[4] Schmidt, M., and Lipson, H. Distilling free-form natural laws from experimental data. *science 324*, 5923 (2009), 81–85.

[5] Szederkényi, G., Banga, J. R., and Alonso, A. A. Inference of complex biological networks: distinguishability issues and optimization-based solutions. *BMC Systems Biology 5* (2011), 177.

[6] Tuza, Z. A., and Stan, G.-B. Characterization of biologically relevant network structures form time series data. In *57th IEEE Conference on Decision and Control, Miami Beach, FL, USA, December 17-19, 2018* (2018).

[7] Tuza, Z. A., and Stan, G.-B. An automatic sparse model estimation method guided by constraints that encode system properties. In *European Control Conference (ECC), Naples Italy* (2019).

[8] Wipf, D., and Nagarajan, S. Iterative reweighted l1 and l2 methods for finding sparse solutions. *IEEE Journal of Selected Topics in Signal Processing 4*, 2 (2010), 317–329.

[9] Ács, B., Szederkényi, G., Tuza, Z., and Tuza, Z. A. Computing all possible graph structures describing linearly conjugate realizations of kinetic systems. *Computer Physics Communications 204* (2016), 11–20.

[10] Ács, B., Szlobodnyik, G., and Szederkényi, G. A computational approach to the structural analysis of uncertain kinetic systems. *Computer Physics Communications 228* (2018), 83–95.

Model-driven design of genetic regulatory networks using virtual parts

Göksel Mısırlı
g.misirli@keele.ac.uk
School of Computing and Mathematics, Keele University

Bill Yang
z.yang22@newcastle.ac.uk
School of Computing, Newcastle University

Anil Wipat
anil.wipat@newcastle.ac.uk
School of Computing, Newcastle University

1 INTRODUCTION

As technology advances, and the cost of DNA synthesis decreases, synthetic biology is moving towards data driven design applications. DNA fragments can be represented as electronic records ready to be composed virtually to create designs of complex genetic circuits. This approach opens the possibility of using representations of various sequence features such as promoters and coding sequences (CDSs) in a computer environment. Whether designs are created manually in a computer-aided environment, or created computationally using heuristic approaches, designs need to be verified and/or optimised. Model-driven design methodologies are already proven as a useful tool to map a virtual design to a physical system in a number of industries including software engineering, aerospace industry, and embedded system development. This approach is also key to create predictable biological applications.

We previously developed the Virtual Parts Repository [4], which provides reusable and modular models of biological parts and interactions. These models are called virtual parts and can be used to create complex models representing biological systems. Due to the increase and the availability of large number of biological parts, virtual parts are ideal to computationally explore large design spaces to create new designs or to find alternative biological solutions.

In parallel to developments in model-driven design approaches, the synthetic biology community developed the Synthetic Biology Open Language (SBOL) [1] to facilitate the exchange of genetic circuit designs. The language allows specifying the order of DNA-based biological parts, their types and interactions between these parts. SBOL is designed as a graph language and can be directly stored in graph repositories. One such database is SynBioHub [3], which allows uploading designs in the form of SBOL documents and querying the underlying data using the SBOL semantics directly. VPR1 uses SBOL only to export information and can only communicate with its built-in relational repository. Model composition is carried out by explicitly querying information about molecular constraints from this repository.

This paper presents the second version of the Virtual Parts Repository (VPR2) in relation to these latest developments. VPR2 has been developed using a modular architecture including components for a Web-based repository, a web service, a client library for computational tools, and a standalone data library to retrieve data from remote SBOL repositories (Figure 1). The web service can be used to retrieve virtual parts, and to create computational models of genetic circuits. Moreover, VPR2 can be used to work with a choice of a SynBioHub instance, which can be installed at a different geographical location. VPR2 has a graph-based repository and allows browsing of the underlying data and models. Moreover, SBOL is used as a domain specific language to control the composition of models.

2 THE VIRTUAL PARTS REPOSITORY 2.0 (VPR2)

VPR2 has been designed to decouple the modelling and genetic circuit design processes. Using this approach, design tools can start taking the benefit of computer simulations without delving into details of complex modelling abstractions. The integration between different tools and VPR2 is carried out using already existing and widely adopted data standards in synthetic biology, such as SBOL and the Systems Biology Markup Language (SBML) [2]. VPR2 uses SBOL to specify data that can be converted into modular and hierarchical SBML models, using two approaches:

- Connected mode. VPR2 is connected to a graph repository to create computational models.
- Disconnected mode. VPR2 uses SBOL documents provided by tools to create computational models. Nominal values are used to parameterise modelling entities.

In the connected mode, VPR2 works directly with a graph repository that can store SBOL data. The repository can be the VPR's default repository or can be selected from various SynBioHub instances. A genetic circuit can be represented using SBOL in terms of DNA-based biological parts (Figure 1), either using start and end positions, or using a relative order of parts. VPR2 queries the SBOL repository to retrieve detailed information about biological parts represented in the design.

Computational tools can use VPR2 to return either a detailed SBOL document or an SBML model as explained below:

- A detailed SBOL document. A simplified definition of a genetic circuit is extended with additional constraints from a

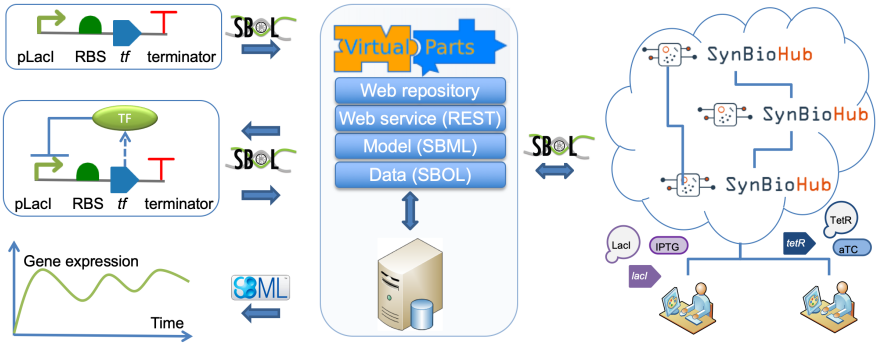


Figure 1: VPR2 has been designed to work with existing standards such as SBOL and SBML. The system can either use data in repositories to return a model for a DNA-based genetic circuit design, or use the information provided by the user to return a model that can be simulated. VPR2 includes a web repository, a web service, model and data layers and a client API.

user-specified SBOL repository. Information about biological constraints such as molecular interactions is used to populate the initial SBOL document. This approach can be used by tools which can create models using custom modelling abstractions and choices of modelling languages.

- An SBML model, capturing the complex relationships between the constituent biological parts.

In the disconnected mode, tools can specify genetic circuits using SBOL with detailed information about molecular interactions between biological parts. This detailed view of a genetic circuit can then be converted into an SBML model.

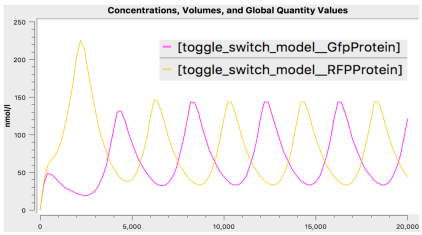


Figure 2: The toggle switch system was modelled using VPR2 and simulated using COPASI.

VPR2 has its own modelling abstraction, representing various biochemical reactions. Examples include the binding and unbinding of biological molecules, transcriptional activation and repression of transcriptional units and the degradation of biological molecules. Virtual parts can include entities for DNA-based parts, proteins and small molecules. Although VPR2 uses nominal values to create modelling entities, tools can override reaction parameters by providing additional details in SBOL documents that are used as input to derive models. Reaction parameters can currently be provided as inline annotations that can be embedded in SBOL documents.

Figure 2 demonstrates the use of VPR2 to model a toggle switch, which was previously modelled using the VPR2 data layer and iBioSim's modelling approach [5]. The aTC and IPTG signals are respectively used to set and unset the GFP and RFP outputs. VPR2 uses a different modelling abstraction utilising the binding and unbinding of molecules to facilitate modularity. In this paper, the system modelled using virtual parts corresponds to 49 biological entities such as promoters, CDSs, proteins and signalling molecules. The resulting model consists of 34 submodels, including 50 biological reactions and 74 different kinetic rate parameters. Submodels can represent virtual parts or templates that are used to instantiate virtual parts.

3 CONCLUSION

VPR2 has been developed to facilitate the search of large design spaces of biological systems using computer simulations. The use of a modular approach and the adoption of existing standards make VPR2 ideal for genetic design automation related workflows. Design tools can, hence, take the advantage of mathematical models to find optimum solutions.

Availability. The VPR2 development version is available at http://v2.virtualparts.org. Please see the *Documentation* menu to access the web service, the web service client library, examples and the data library.

REFERENCES

[1] Cox, R. S., et al. Synthetic biology open language (SBOL) version 2.2.0. *J Integr Bioinform 15*, 1 (2018).

[2] Hucka, M., et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics 19*, 4 (2003), 524–531.

[3] McLaughlin, J. A., et al. SynBioHub: A standards-enabled design repository for synthetic biology. *ACS Synth Biol 7*, 2 (2018), 682–688.

[4] Misirli, G., et al. Composable modular models for synthetic biology. *ACM J Emerg Technol Comput Syst 11*, 3 (2014), 22.

[5] Misirli, G., Nguyen, T., et al. A computational workflow for the automated generation of models of genetic designs. *ACS Synth Biol* (2018).

Stochastic Analysis of an Genetic Sensor

Jeanet Mante
University of Utah
Salt Lake City, UT, USA
jv@mante.net

Pedro Fontanarrosa
University of Utah
Salt Lake City, UT, USA
pfontanarrosa@gmail.com

Chris Myers
University of Utah
Salt Lake City, UT, USA
myers@ece.utah.edu

1 INTRODUCTION

The field of synthetic biology has developed with the aim of making genetic engineering more repeatable and reproducible. Part of the synthetic biology movement is the creation of genetic circuits which carry out useful functions such as sensing environmental conditions or producing useful pharmaceuticals. To make the design of such circuits easier, inspired by digital electronic design, a more abstract view of the components is taken in order to allow composing of parts together to create a desired functionality. An example of this is seen with software tool, Cello, which converts specifications of behavior in the Verilog language into a composition of genetic parts [3].

Unfortunately, the analogy to digital electronic circuits breaks down when the noise of biological systems is considered, and thus further modeling and verification is required to check designs that are generated via automated methods. This is because unlike the use of transistor based logic in which an abundance of electrons create highly predictable and binary behavior, genetic circuits involve small molecule counts making their behavior inherently noisy and stochastic. The fact that they are actually many circuits running in parallel within a cellular population, further complicates the analysis of these circuits. Therefore, stochastic analysis methods are required to understand the noise tolerance (i.e. robustness) of these circuits. As a case study, this paper considers the stochastic analysis of an asynchronous genetic sensor circuit, building on the case study in [2]. This design is particularly interesting, as it is constructed using three different cell types, and its correctness depends upon the interactions between these cells.

2 RESULTS

The genetic sensor circuit for our case study is shown in Figure 1, and it was originally proposed in [2]. The desired behavior of this circuit is the filtering out of signals that are only present for a short amount of time. Namely, this circuit is triggered when it senses the presence of both IPTG and aTc, and IPTG must be present in the system for a prolonged period of time to propegate through the three levels of logic. If this happens, it produces the output signal *yellow fluorescent protein* (YFP). A circuit with this behaviour may be potentially interesting for applications such as cancer treatments where a drug should only be released if tissue specific

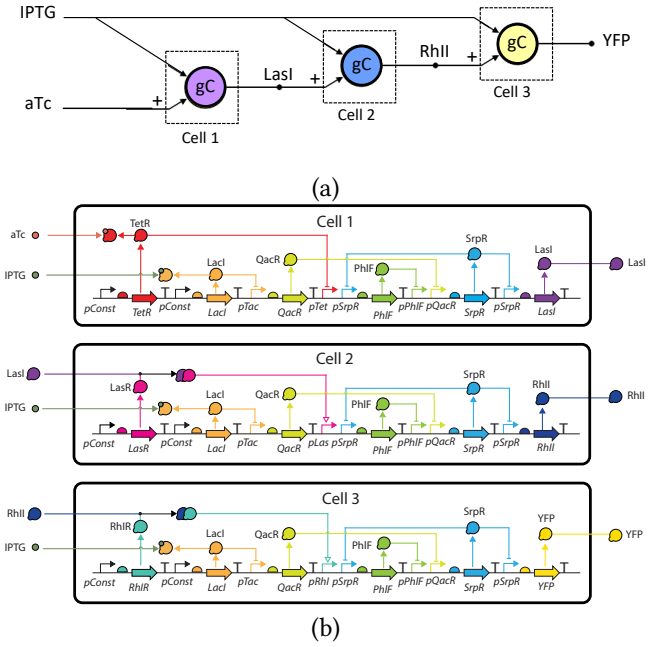


Figure 1: A genetic sensor that uses filtering and communication to improve its reliability. (a) The logic diagram produced by logic synthesis. It is composed of three gC gates which go high when both inputs are high and go low when the input not marked with a "+" goes low. The output of the second and third gate are connected to the "+" input of the next gate. The detection begins when both IPTG and aTc go high, activating Cell 1. This creates the quorum signal LasI to diffuse to Cell 2, which then activates Cell 2 to produce the quorum signal RhlI. The RhlI signal diffuses to Cell 3 to activate YFP production. However, if IPTG goes low at any point during this chain reaction, the whole circuit resets. (b) The genetic design produced by Cello, which is composed of three genetic sequences that can be put onto separate plasmids and transformed into cells to create three cell types. (Figure courtesy of [2])

molecules were sensed and tumour indicator molecules are continuously present, thus acting analogous to a low pass filter.

As described earlier, in order to determine if this circuit has the desired behavior, a stochastic analysis must be performed. In [2], an ODE analysis shows that this circuit potentially has

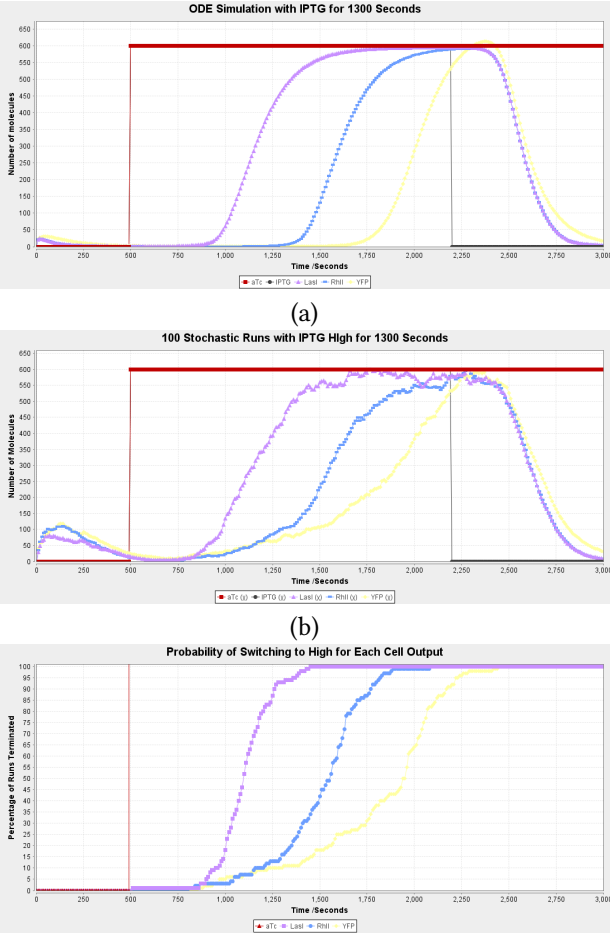


Figure 2: iBioSim simulations of the genetic sensor demonstrating the consistency of the behavior. For the circuit outputs to turn on (Purple, Blue, and Yellow High) both aTc and IPTG inputs need to be present for a sufficient amount of time (as it filters lower time presence). (a) The ODE Simulation with IPTG present for 1300 seconds. This shows that all cell outputs (Purple, Blue, and Yellow) are high and that a time of 1300 seconds is enough for the final output (YFP) to be reach a high state, and that there is a delay of about 400 seconds between LasI reaching High Equilibrium and YFP doing the same. (b) The stochastic simulation with IPTG present for 1300 seconds. This shows good agreement with the ODE analysis. (c) Results from 100 Monte Carlo simulation runs showing the probability LasI, RhlI, and YFP going above 300 molecules versus time. This shows variation in timing of the response of cells in a population.

the desired behavior (see Figure 2(a)). This paper conducts a stochastic analysis to better understand the robustness of this circuit in the presence of noise. The results of this stochastic analysis are shown in Figure 2(b) and (c). Figure 2(b) shows the average of 100 stochastic simulation runs, and there is good agreement with the behavior seen in the ODE analysis. Figure 2(c) shows the probability of LasI, RhlI, and YFP going above 300 molecules versus time during these 100 stochastic simulation runs. These results give a clear indication that not all cells make these transitions at the same time. In other words, some genetic sensors will respond to the inputs faster than others will respond slower. It can also be seen that noise is cumulative between cells.

3 METHODS

The genetic circuit was designed using the software tools Cello [3] and iBioSim [4] as described in [2]. A model was created from this genetic circuit using the model generation procedure described in [1]. The simulations were conducted using ODE and stochastic simulators in iBioSim.

4 DISCUSSION

This paper gives some initial results on the stochastic analysis of a complex genetic circuit. *Asynchronous sequential circuits* (circuits whose behavior depends on the timing and order in which the inputs are applied) such as the one studied here require complicated analysis. Furthermore, this system is composed of three cell types each with a different genetic circuit that together communicate to perform the desired filtering behavior. Given the inherently stochastic nature of genetic circuits, a stochastic analysis is necessary to determine the robustness of such genetic designs. Our preliminary stochastic results indicate that the circuit behaves as desired. In the future, we would like to perform a more detailed analysis using techniques such as stochastic model checking.

Acknowledgements

This work is supported by the National Science Foundation (CCF-1748200) and DARPA (FA8750-17-C-0229). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Misirli, G., Nguyen, T., McLaughlin, J. A., Vaidyanathan, P., Jones, T. S., Densmore, D., Myers, C., and Wipat, A. A computational workflow for the automated generation of models of genetic designs. *ACS Synthetic Biology* (2018). PMID: 29782151.

[2] Nguyen, T., Jones, T., Fontanarrosa, P., Mante, J., Zundel, Z., Densmore, D., and Myers, C. Design of asynchronous genetic circuits. *Proceedings of the IEEE* (2019).

[3] Nielsen, A. A. K., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. Genetic circuit design automation. *Science 352*, 6281 (2016).

[4] Watanabe, L., Nguyen, T., Zhang, M., Zundel, Z., Zhang, Z., Madsen, C., Roehner, N., and Myers, C. ibiosim 3: A tool for model-based genetic circuit design. *ACS Synthetic Biology* (2018). PMID: 29944839.

Visualization of Part Use in SynBioHub

Jeanet Mante
University of Utah
Salt Lake City, USA
j.mante@utah.edu

Zach Zundel
University of Utah
Salt Lake City, USA
me@zachzundel.com

Chris Myers
University of Utah
Salt Lake City, USA
myers@ece.utah.edu

1 INTRODUCTION

Synthetic biology is a movement to standardise genetic engineering and make it more repeatable. One important advancement was the development of standardised genetic parts known as *BioBricks*, which can be composed using restriction enzyme assembly [3, 5]. Another important advancement was the development of the *Synthetic Biology Open Language* (SBOL), a standard language for describing these parts, among other things [1]. Finally, to share these parts, design repositories, e.g. SynBioHub [4], were developed. One recent enhancement to SynBioHub is the introduction of plugin capabilities to allow third-party developers to add functionality. This paper describes one such plugin that is designed to aid designers in part selection. In particular, this plugin enables users considering the use of a particular part to see alternative parts with the same function that have been heavily utilized in the past, or to see parts that are commonly used with the part of interest, enabling users to find additional parts for their genetic circuit design. As a proof-of-concept, this plugin is demonstrated using a library of BioBrick parts from the *International Genetically Engineering Machine* (iGEM) competition.

2 BACKGROUND

A genetic design is the combination of one or more genetic parts, in series. There are many different possible part categories, however the four we focused on (for their prevalence and importance) are promoters, ribosome binding sites (RBS), coding sequences (CDS), and terminators, (other roles are also displayed in a category called Other). Within each category there are many different possible parts to choose, e.g. for promoters pTet and pLac are commonly used.

When creating a new design, either old parts can be reused, or one can create their own new parts with associated DNA sequences. An issue with this is that for something like the pTet promoter slight variations in naming, sequences, or upload file format can lead to a whole host of parts which are the same or very similar. Thus, it can be difficult to tell which one is the most common form, which often results in further new parts being uploaded.

3 RESULTS

Our visualization plugin can assist designers to find parts for their designs. In particular, our plugin displays up to three graphs per part page. First, there is a histogram that shows how commonly used this part is relative to most commonly used parts (not shown). Second, there is a histogram that shows the commonality of this part relative to other commonly used parts of the same role (see Figure 1). Finally, there is a Sankey diagram that shows other parts that are commonly used with this part sorted by their role and indicating their position compared to the part of interest (see Figure 2). Some additional features that make these visualizations interactive include the ability to: zoom in and out, scroll across the graphs, show data about a part when hovering over it, add value comparison lines dynamically, download a plot as a PNG, and navigate to a page describing that part in the corresponding SynBioHub instance.

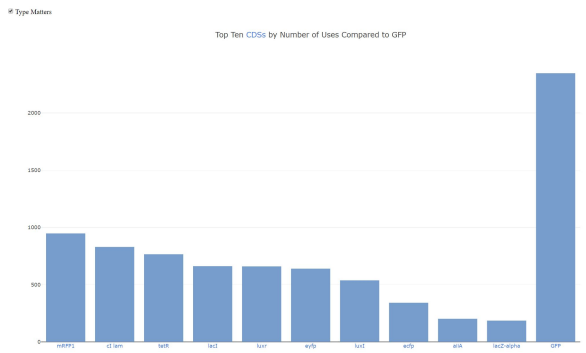


Figure 1: Top 10 CDSs by Number of Uses Compared to GFP.

4 METHODS

The visualization tool is implemented as a plugin deployed on the development server for the reference instance of SynBioHub (https://dev.synbiohub.org). The plugin is an HTTP server, written in Python, using the Flask library with endpoints for each of the graph types. When a part page is opened, SynBioHub sends a post request to this plugin with the URL for the part. This URL is used in four different pre-written SPARQL queries which are sent to SynBioHub to gather the necessary data. The data returned is processed to create the input for the visualizations. The visualizations are made using the plotly [2] library. Finally, the graphics are transmitted to SynBioHub for rendering on a part page in SVG format embedded in HTML. Repo: https://github.com/3ach/jet-plugin

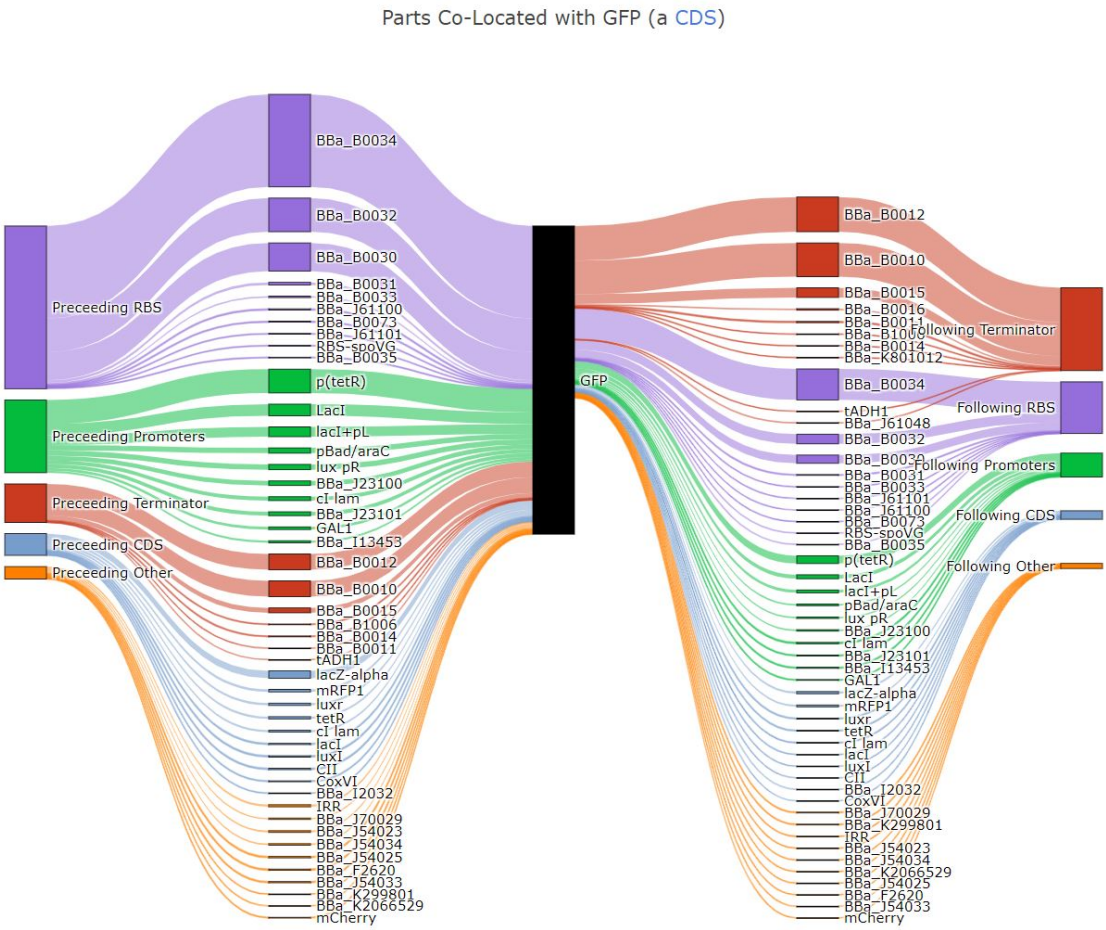


Figure 2: Example visualizations created by the plugin for the part BBa_E0040, more commonly known as GFP. A diagram showing how the part of interest (GFP) is combined with other parts in the designs it is found in and what fraction of these interactions are preceding (e.g. BBa_B0034 is before GFP) and following (e.g. BBa_B0034 is after GFP).

5 DISCUSSION

These visualisations allow easier understanding of the data in SynBioHub and allow questions to be answered such as:

- What role of part is most often seen in combination with a part? (RBS are most commonly seen with GFP)
- Which part is most often seen in combination with a part? (BBa_B0034 for GFP)
- Does a particular part generally precede or follow GFP? (BBa_B0034 generally precedes GFP)
- Does a part generally occur before or after other parts? (After for GFP)

In the future, we plan to incorporate features like more precise ordering information and sequence similarity analysis.

Acknowledgements

This work is supported by the National Science Foundation (CCF-1748200) and DARPA (FA8750-17-C-0229). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Galdzicki, M., Clancy, K. P., Oberortner, E., Pocock, M., Quinn, J. Y., Rodriguez, C. A., Roehner, N., Wilson, M. L., Adam, L., Anderson, J. C., and et al. The synthetic biology open language (sbol) provides a community standard for communicating designs in synthetic biology. *Nature Biotechnology 32*, 6 (Jun 2014), 545–550.

[2] Inc., P. T. Collaborative data science, 2015.

[3] Knight, T. Idempotent vector design for standard assembly of biobricks. Tech. rep., MIT Artificial Intelligence Laboratory; MIT Synthetic Biology Working Group, 2003.

[4] McLaughlin, J. A., Myers, C. J., Zundel, Z., Misirli, G., Zhang, M., Ofiteru, I. D., Goni-Moreno, A., and Wipat, A. Synbiohub: A standards-enabled design repository for synthetic biology. *ACS Synthetic Biology 7*, 2 (Feb 2018), 682–688.

[5] Shetty, R. P., Endy, D., and Knight, T. F. Engineering biobrick vectors from biobrick parts. *Journal of Biological Engineering 2*, 1 (Apr 2008), 5.

SBOL Visual 2 Ontology

Göksel Mısırlı[1], Jacob Beal[2], Thomas E. Gorochowski[3], Guy-Bart Stan[4], Anil Wipat[5], Chris Myers[6]
[1]Keele University, UK, [2]Raytheon BBN Technologies, USA, [3]University of Bristol, UK, [4]Imperial College London, UK,
[5]Newcastle University, UK, [6]University of Utah, USA
g.misirli@keele.ac.uk,jakebeal@ieee.org,thomas.gorochowski@bristol.ac.uk,g.stan@imperial.ac.uk,
a.wipat@newcastle.ac.uk,myers@ece.utah.edu

1 INTRODUCTION

The ability to visually represent genetic circuits can aid human understanding and improve the dissemination of information. Such visual representations are especially useful in publications for helping to explain complex relationships between constituent parts of large genetic circuits. However, when these diagrams are created manually, variations in how information is presented may cause issues in interpreting the meaning of different glyphs and how they are connected. Moreover, there are an increasing number of computational tools and repositories for synthetic biology that can automatically render visual depictions of genetic circuits. The SBOL Visual [2] standard has been developed to provide guidelines on how genetic design features should map to suitable glyphs, as well as how various glyphs can be connected together. However, an actual computational implementation of mapping genetic parts to suitable glyphs has previously been left to tool developers.

The SBOL Visual community has created a set of standard glyphs for a variety of commonly used genetic parts. Glyphs are proposed by members of the community and, through discussions, decisions are made about the types of part each can be used to represent. In addition to recommended glyphs for specific types of part, SBOL Visual specifies generic and alternative glyphs for many. These annotations are available as free text. For each genetic part type, a single human-editable Markdown file is created, which includes information about the mapping between recommended and alternative glyphs, and the relevant genetic parts through biological roles and molecular interactions, which are identified via commonly used ontological terms.

Ontological terms are a powerful way to represent a large amount of information via simple URIs, which can then further point to additional properties of terms and the relationships of terms with other biological concepts. Furthermore, the meaning of a term is also derived from all its parents. As a result, an ontological representation of SBOL Visual is highly desirable for computational integration and processing of visual guidelines with other existing ontologies and tools. Previously, an ontology was created for SBOL 1 and included mappings only between DNA-based parts and SO terms [8]. Since then SBOL has grown into a richer data model, with many more glyphs defined, as well as new classes of glyphs and relationships between glyphs. Thus, the ontology needed to be reconstructed in the light of these developments.

To address this, we have developed the SBOL Visual 2 Ontology, which we use to represent the constraints about genetic circuit glyphs and their relationships to other ontological terms. Recently, an ontology called SBOL-OWL [7] was developed to provide semantic meaning for terms from the SBOL standard [3] in a machine accessible format. Using an ontological mappping, the SBOL Visual Ontology further integrates information about standardized glyphs with the SBOL standard.

2 THE SBOL VISUAL ONTOLOGY

The SBOL Visual 2 Ontology (SBOL-VO) was programmatically constructed using Markdown files that are created and managed by the SBOL Visual community. The base class in the ontology is `Glyph`, a subclass of which corresponds to an individual glyph.

A class representing a glyph may include the following `Annotation` properties: `rdfs:label` (name), `rdfs:comment` (description), `defaultGlyph` (the name of the glyph file), `glyphDirectory` (the folder containing the glyph), `notes` (additional free text information), `recommended` (whether the glyph is recommended or not), `prototypicalExample` (an example use of the glyph).

SBOL-VO was directly integrated with the SBOL standard via SBOL-OWL through ontological restrictions. These restrictions are created based on ontological terms associated with a glyph. For example `AptamerGlyph` is defined to be a glyph for `sbol:ComponentDefinition` entities with the `role` of `SO:0000031` which is a Sequence Ontology (SO) [5] term used for aptamers. These restrictions can be defined for `ComponentDefinition` entities that represent genetic parts or `sbol:Interaction` entities that represent molecular interactions. The following rules were applied to create these restrictions:

- If a glyph is associated with an SO term, a restriction is created for the `ComponentDefinition` entity using the `role` property. For example, AptamerGlyph `isGlyphOf` some `ComponentDefinition` with a `role` of SO:0000031.
- If a glyph is associated with a BioPAX [4] term, the restriction is created for the `ComponentDefinition` entity using the `type` property. For example, ComplexGlyph

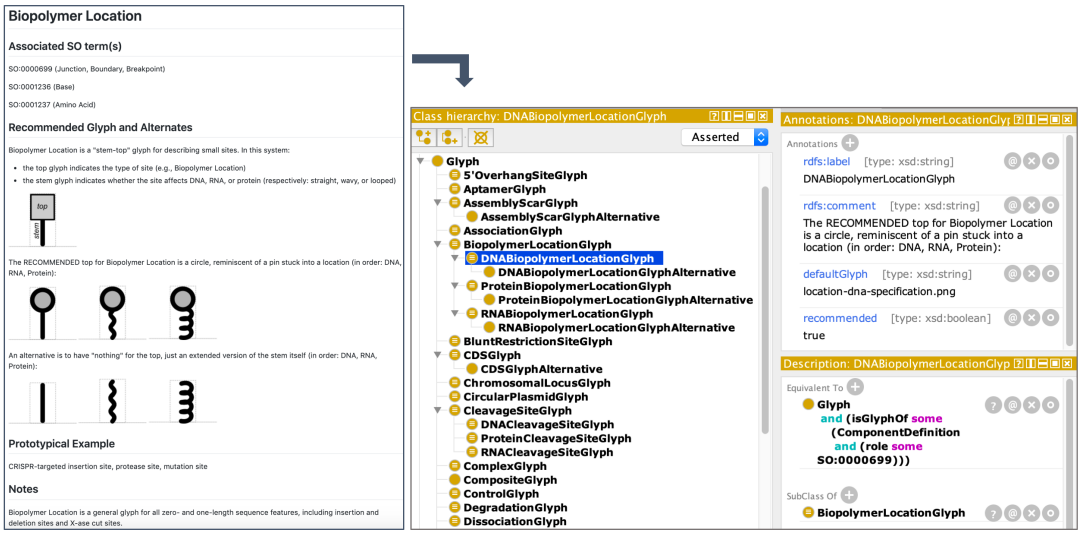


Figure 1: Free text description of recommended and alternative glyphs are used to create ontology terms and restrictions.

`isGlyphOf` some `ComponentDefinition` with a type of `biopax:Complex`.

- If a glyph is associated with a Systems Biology Ontology (SBO) [1] term, a restriction is created for the `Interaction` entity using the `type` property, only if the SBO term is a subclass of the biological activities and processes (i.e., `SBO:0000231`). For example, DegradationGlyph `isGlyphOf` some `Interaction` with a `type` of `SBO:0000179`.

Based on the representation of information in the Markdown files, hierarchical relationships between SBOL Visual terms were also created. The following rules were applied to create parent-child relationships.

- If a single glyph is included, the corresponding term is created, e.g. AptamerGlyph.
- If both a recommended glyph and an alternative glyph are included, the mapping restriction is created for the recommended term. The alternative glyph term is created as a subclass of the former and linked to the former via the `isAlternativeOf` property, e.g. AssemblyScarGlyph and AssemblyScarGlyphAlternative terms.
- If one generic glyph and a set of its instances (n glyphs) are included, the base class is created for the former and one recommended term is created for each instance, e.g. CleavageSiteGlyph is the parent for terms about DNA, Protein and RNA cleavege sites. If alternatives are included, they are created as subterms of the recommended terms, e.g. BiopolymerLocationGlyph (Figure 1) and its recommended and alternative terms.

The programmatic conversion was carried out using the Python programming language and using the OWLready API [6]. The ontology is available at https://dissys.github.io/sbol-visual-ontology.

3 CONCLUSION

SBOL-VO makes standard glyphs used for genetic circuit diagrams available to computational tools in the form of an ontology. The SBOL community heavily uses ontological terms to map genetic parts and their roles. Here, the creation of SBOL-VO and its mapping with the SBOL ontology facilitates further data integration for querying and retrieval of appropriate glyphs for genetic parts and their interactions.

4 ACKNOWLEDGEMENTS

Funded in part by NSF Expeditions in Computing Program Award #1522074 as part of the Living Computing Project. This document does not contain technology or technical data controlled under either U.S. International Traffic in Arms Regulation or U.S. Export Administration Regulations.

REFERENCES

[1] Courtot, M., et al. Controlled vocabularies and semantics in systems biology. *Molecular systems biology 7*, 1 (2011), 543.
[2] Cox, R. S., , et al. Synthetic biology open language visual (SBOL visual) version 2.0. *Journal of integrative bioinformatics 15*, 1 (2018).
[3] Cox, R. S., et al. Synthetic biology open language (SBOL) version 2.2.0. *Journal of integrative bioinformatics 15*, 1 (2018).
[4] Demir, E., et al. The BioPAX community standard for pathway data sharing. *Nature biotechnology 28*, 9 (2010), 935.
[5] Eilbeck, K., et al. The sequence ontology: a tool for the unification of genome annotations. *Genome biology 6*, 5 (2005), R44.
[6] Lamy, J.-B. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine 80* (2017), 11–28.
[7] Misirli, G., et al. SBOL-OWL: An ontological approach for formal and semantic representation of synthetic biology information. *ACS Synthetic Biology* (2019).
[8] Quinn, J., et al. Synthetic biology open language visual: an ontological use case. *Extended abstract at Bio-Ontologies* (2013).

Flapjack: an open-source tool for storing, visualising, analysing and modelling kinetic gene expression data

Guillermo Yáñez
Pontificia Universidad Católica de Chile
gnyanez@uc.cl

Isaac Núñez
Pontificia Universidad Católica de Chile
innunez@uc.cl

Tamara Matute
Pontificia Universidad Católica de Chile
tfmatute@uc.cl

Fernán Federici
Pontificia Universidad Católica de Chile
ffederici@bio.puc.cl

Timothy J. Rudge
Pontificia Universidad Católica de Chile
trudge@uc.cl

1 INTRODUCTION

Engineering design cycles based on accurate parameter estimation from experimental data are key for predictable assembly of complex genetic circuits. In particular, as dynamical systems the reliable design of genetic circuits requires analysis of kinetic gene expression data. Metadata not present in raw data files is also necessary to account for experimental protocols and the context in which measurements were made. This data is often distributed across many institutions, in different file formats, repositories and databases, which makes it difficult to collate and reduces the power of analysis. Existing repositories focus on biological part descriptions [1], parameter estimation from published data [2], non-kinetic -omics experimental data [3], or store data as raw files for individual experiments [4]. None of these repositories uses the widely adopted Synthetic Biology Open Language (SBOL) standard [5] for description of genetic parts, making it difficult to relate data on genetic circuits to their constituent components. Thus there is a need for data repositories that can store kinetic gene expression data, link this data to circuit designs, and allow analysis that combines multiple studies and experiments to reliably estimate parameters.

2 RESULTS

Here we present Flapjack, a web-based open-source tool for storing, visualising, analysing and modelling kinetic gene expression data. Flapjack is an SBOL-compliant full-stack web application built with the back-end framework Django, using Python for analysing data via libraries including NumPy, SciPy and Pandas. The front-end is written in JavaScript and uses the D3.js visualisation library. The architecture of Flapjack is shown in Figure 1. We provide a web app interface, and a Docker container for easy installation at individual, lab or institution scale. Users are able to upload their raw measurement data and metadata, linking the circuit DNA to SynBioHub Uniform Resource Identifiers (URIs), and describing experimental conditions such as media, strain, and inducer

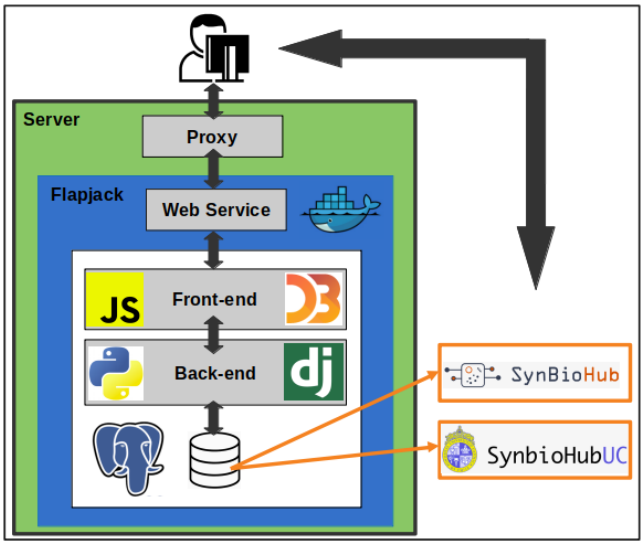


Figure 1: Flapjack's stack: PostgreSQL relational database; Back-end Django Framework; Front-end; D3.js library. All contained in a Docker virtualization.

concentrations. The uploaded information is then stored in a PostgreSQL relational database (Figure 2). The application back-end communicates with the database through Django's Object-Relational Mapping (ORM) Framework. Flapjack is also available as a python module for developers to use with their own custom software tools and a local database. Flapjack enables users to flexibly query experimental data based on metadata, for example extracting all measurements related to a particular DNA sequence, all growth curves for a given strain, etc. These queries return all relevant data irrespective of the particular study or experiment, or may be restricted to specific experiments of interest. Using the web app users may then visualize the experimental time series using interactive plots (time courses, kymographs, heatmaps), apply analyses such as calculation of expression

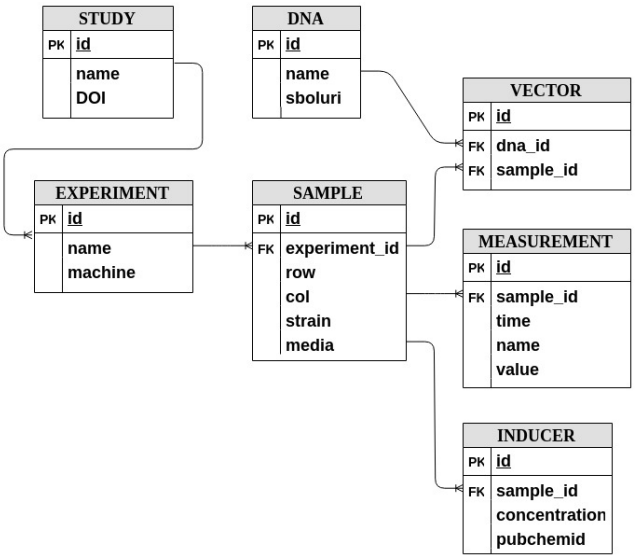


Figure 2: Flapjack's data model.

rates, growth rates, and parameterise transfer functions or induction curves (Figure 3). For custom analysis queried data can be downloaded in CSV, JSON or XML file formats.

We demonstrate the functionality of Flapjack by characterizing six multifluorescent gene circuits: constitutive expression, IPTG signal receiver input module, and four inverters [6] with IPTG input. Each circuit was measured in two separate experiments, with two replicates in each experimental condition. Inverters were measured under 12 different inducer concentrations. Circuit designs were made using SBOLDesigner [7] and uploaded to our SynBioHub [8] instance [9]. Figure 3 shows examples of visualisation outputs from the web app, analyses of kinetic gene expression rates, and parameter estimation of induction curves.

Currently, Flapjack supports upload of Synergy HTX and BMG Labtech microplate reader data, but can easily be extended to accommodate any data format.

3 FUTURE WORK

We are currently developing data federation via *Resource Description Framework* (RDF) and the implementation of a Virtuoso database; developing a Flapjack API for developers to retrieve data from Flapjack and use it directly in their own software tools without the need for file download; including upload of data from sources such as timelapse single-cell microscopy; and coupling of models [10] to synthetic microbial biofilm simulation tools such as CellModeller [11].

4 CONCLUSIONS

We have developed a data repository and set of analysis tools that enables scaleable data analysis, visualization and parameter estimation, collating data between institutions, studies, experiments, and individual researchers. Our tool could significantly decrease time consuming and error prone data wrangling, allowing characterization of genetic circuits that seamlessly incorporates new data for reliable parameter estimation. Through the use of the SBOL Stack, our tool connects directly to the ecosystem of existing tools such as SBOLDesigner, SynbioHub, and iBioSim. Flapjack could significantly enhance data sharing, management and analysis for synthetic and systems biology.

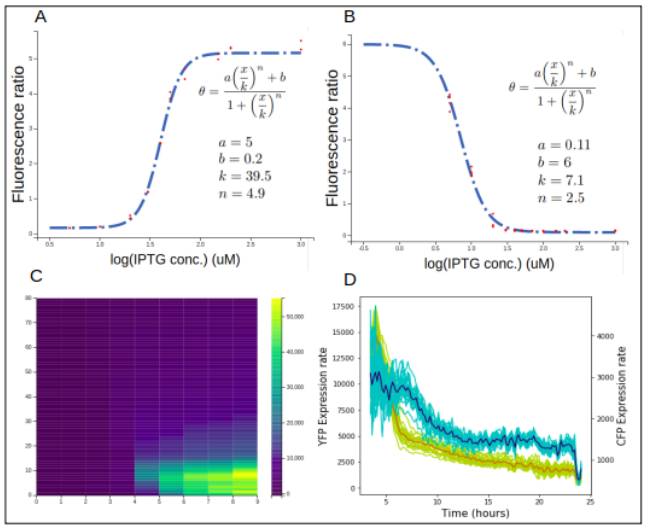


Figure 3: Flapjack's analysis. A. Input curve for IPTG signal receiver. B. QacR Inverter with IPTG input. C. Kymograph for YFP Expression rate. D. Expression rate for YFP and CFP

REFERENCES

[1] igem parts repository. https://igem.org/Registry.
[2] Linh Huynh and Ilias Tagkopoulos. A Parts Database with Consensus Parameter Estimation for Synthetic Circuit Design. *ACS Synth. Biol.*, 5(12):1412–1420, 2016.
[3] William C. Morrell et al. The Experiment Data Depot: A Web-Based Software Tool for Biological Experimental Data Storage, Sharing, and Visualization. *ACS Synth. Biol.*, 6(12):2248–2259, 2017.
[4] Biodare. https://www.biodare.ed.ac.uk.
[5] Michal Galdzicki et al. The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nature Biotechnology.*, 32:545–550, 2014.
[6] Alec A. K. Nielsen et al. Genetic circuit design automation. *Science*, 352(6281):53–64, 2016.
[7] Michael Zhang et al. SBOLDesigner 2: An Intuitive Tool for Structural Genetic Design. *ACS Synth. Biol.*, 6(7):1150–1160, 2017.
[8] James A. McLaughlin. SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology. *ACS Synth. Biol.*, 7(2):682–688, 2018.
[9] Synbiohubuc. https://synbio.ing.puc.cl/synbiohub.
[10] Timothy S. Gardner et al. Construction of a genetic toggle switch in Escherichia coli. *Nature.*, 403:339–342, 2012.
[11] Timothy J. Rudge et al. Computational Modeling of Synthetic Microbial Biofilms. *ACS Synth. Biol.*, 1(8):345–352, 2012.

A Reconfigurable Digital Microfluidics Platform

Georgi Tanev[1,2], **Luca Pezzarossa**[1], **Winnie E. Svendsen**[2], **and Jan Madsen**[1]
[1]Department of Applied Mathematics and Computer Science
[2]Department of Biotechnology and Biomedicine
Technical University of Denmark
Kongens Lyngby, Denmark
{geta,lpez,wisv,jama}@dtu.dk

1 MANIPULATING BIOLOGICAL COMPONENTS

Constructing or modifying already existing biological systems is the nucleus of the synthetic biology field. Engineering of novel biosensors, reprogramming living cells, creating genetic logic circuits for biological computers, and engineering new synthetic life organisms are some of the most remarkable cross-disciplinary endeavours in recent years. Designing, assembling, and manipulating biological components are core aspects of the synthetic biology field but they are also a bottleneck in the typical research design-build-test cycle. Computer-aided design tools and hardware instruments are used to automate, reduce complexity, and accelerate the computational and experimental work in the research domain. Furthermore, an increasing number of domain-specific computer-aided tools for design, programming, and simulation have been standardized, developed, and adopted by the synthetic biology community. By contrast, the physical manipulation of biological samples still heavily relies on expensive traditional wet lab equipment such as pipetting robots, specialized thermal cyclers, and measuring instruments. The goal of our work is to develop a reconfigurable biochip alternative to the traditional lab liquid handling by utilizing lab-on-a-chip technologies. In this paper, we present a modular digital microfluidics platform which has the potential to integrate a broad panel of lab processes used when working with biological components.

2 DIGITAL MICROREACTORS

Precise and efficient liquid handling in a controlled environment is an essential part of the process of manipulating biological samples. Ensuring reagent concentration, temperature, mixing order, incubation, purification, sensing, and analysis are in the foundation of accurate and reproducible experimental lab work. Conducting a lab experiment often involves moving reagents between storage containers, reaction chambers or lab equipment and it is a tedious and error-prone task. Lab robots can be used as an alternative but due to their cost and operation complexity, they are more suitable for large scale automation. As an alternative, a fluid handling technology called digital microfluidics provides a simple and efficient way of moving nano-to-microliter droplets on a biochip patterned with electrodes. Discrete droplets can be programmatically dispensed from on-chip reservoirs, transported, merged, mixed, and split. Each droplet serves the purpose of a fluid vehicle and a microreactor thus eliminating the need for storage containers and mechanical fluid handling. Integrating a biochip with an array of sensing and controls allows for a universal full stack solution for working with biological systems.

3 A RECONFIGURABLE BIOCHIP PLATFORM

A reconfigurable biochip and its instrumentation should be able to adapt and evolve together with the experimental needs. To address this, we present the latest evolution of our modular reconfigurable digital microfluidics platform [3] the architecture of which is shown in Figure 1.

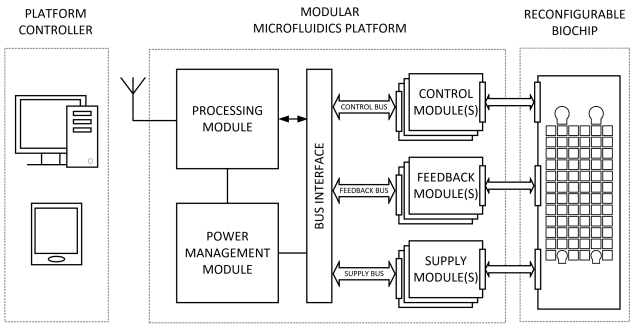


Figure 1: Modular reconfigurable microfluidics platform architecture.

The platform is designed with respect to modularity on both software and hardware levels and it consists of three distinctive parts: (1) a computer or a smart device which acts as a high-level programming tool and platform controller, (2) an instance of the modular microfluidics platform, and (3) a reconfigurable biochip. Key feature of the platform are the clear separation between the high-level and low-level computation functions as well as the loosely coupled modular instrumentation subsystem. The *bus interface* provides a plug-and-play connection to the *control*, *feedback*, and *supply* modules used to instrument a particular biochip. The plug-and-play functionality allows for new platform modules to be developed and added in order to match any changes in

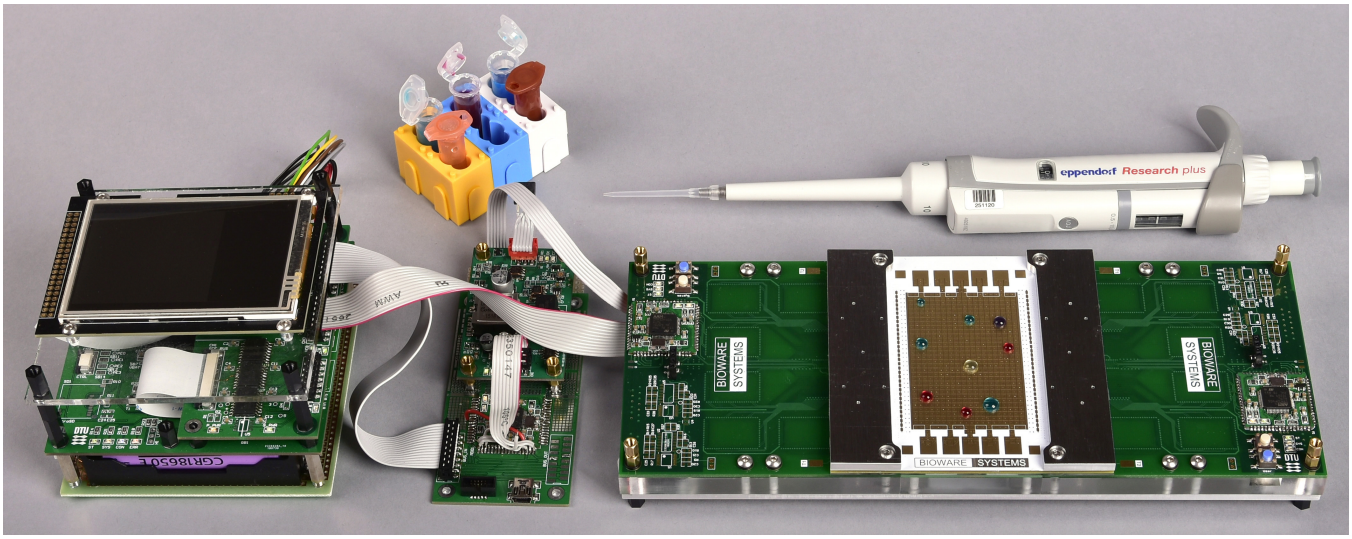


Figure 2: Modular microlfuidics platform prototype.

the instrumentation needs of a biochip. Furthermore, the modular design enables the microfluidics platform to be represented as a library of functional modules and to be used as an input to a high-level protocol compiler. This allows for a biological protocol to be compiled into a biochip design and a corresponding instance of the microfluidics platform.

A demo configuration of the instrumentation platform with an open digital biochip is shown in Figure 2. A central part of the system is the replaceable biochip which is fabricated by commercially available PCB processes. The chip has a functional array of individually addressable electrodes and reconfigurable heating regions. Implementing cooling capabilities for reagent storage or thermal shock as well as providing means for electrophoresis and electroporation are considered as future work.

4 ON THE BIOCHIP HORIZON

Similar DMF devices and platforms have been previously reported [1][2][4] and each of them has a different level of integration, programming model, complexity, modularity, and workflow. Digital microfluidic biochips are built around an array of individually addressed electrodes. This allows for time multiplexing of different specialized regions of the chip in order to achieve optimal resources utilization. Such resources can be temperature controlled zones, biosensors or simply clean or contaminated areas of the biochip. Another important characteristic of the digital biochips is that they allow for an in-line integration of detectors to provide intermittent or continuous monitoring of the ongoing reactions.

Automating and miniaturizing DNA assembly, amplification and transformation on low-cost portable devices hold the potential to reduce operation cost and increase the throughput of experimental validation of the synthetic biology research. Biochips are becoming more features rich, integrated, and simpler to fabricate. Providing modularity and standardization on biochip component, instrumentation and programming level is considered to be a key factor of sustaining the fast development pace as well as lowering the barrier of entering the biochip research field.

5 ACKNOWLEDGMENTS

This research was supported by the *Copenhagen Center for Health Technology* (CACHET) and the *Novo Nordisk Foundation (NNF)* grant NNF18OC0034210.

REFERENCES

[1] Alistar, M., and Gaudenz, U. OpenDrop: An Integrated Do-It-Yourself Platform for Personal Use of Biochips. *Bioengineering 4*, 4 (2017), 45.

[2] Fobel, R., Fobel, C., and Wheeler, A. R. DropBot: An open-source digital microfluidic control system with precise control of electrostatic driving force and instantaneous drop velocity measurement. *Applied Physics Letters 102*, 19 (2013).

[3] Tanev, G., Svendsen, W., and Madsen, J. A modular reconfigurable digital microfluidics platform. *Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS, DTIP 2018* (2018), 1–6.

[4] Willsey, M., Stephenson, A. P., Takahashi, C., Vaid, P., Nguyen, B. H., Piszczek, M., Betts, C., Newman, S., Joshi, S., Strauss, K., and Ceze, L. Puddle: A dynamic, error-correcting, full-stack microfluidics platform. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (2019), ASPLOS '19, ACM, pp. 183–197.

Design Automation of Microfluidic Droplet Sorting Platforms

David McIntyre
Department of Biomedical Engineering
Boston University
dpmc@bu.edu

Douglas Densmore*
Department of Electrical and Computer Engineering
Boston University
dougd@bu.edu

1 INTRODUCTION

Both basic research and biological design require high throughput screening to parse through the massive amounts of variants generated in experiments. However, the cost and expertise needed for use of such technology limit accessibility. Simple and reproducible designs of a sorting platform would reduce the barrier for implementation of affordable bench-top screening platforms. Droplet microfluidics present a promising approach for automating biology, reducing reaction volumes to picoliter droplets and allowing for deterministic manipulation of samples. Droplet microfluidics have been used extensively for high throughput screening and directed evolution [1, 3], yet limitations in fabrication have prevented the characterization needed for a design tool and subsequent widespread adoption. Here, we present a finite element analysis (FEA) model-based design framework for dielectrophoretic droplet microfluidic sorters and its preliminary experimental validation. This framework extends previous work from our group creating microfluidic designs tools, increasing their usability in the lab [4, 6].

2 FEA MODEL OF DROPLET SORTING

Successful droplet sorting is characterized by deflection of the target droplet from the "waste" to "keep" channel (Figure 1). Total lateral deflection is a result of opposing dielectrophoretic (Eq. 1) and Stokes' drag forces (Eq. 2), where V is the electrode voltage, r_d is the droplet radius, ϵ_{oil} is the oil permittivity, η_{oil} is the oil viscosity, $\vec{v}_y$ is the lateral velocity, and k represents the geometry of the electric field gradient in the y-direction [2]. Droplets are assumed to be solid particles, with the same viscosity and permittivity as water.

$$F_{DEP} = 4k\pi\epsilon_{oil}r_d^3V^2 \tag{1}$$

$$F_D = 6\pi\eta_{oil}r_d\vec{v}_y \tag{2}$$

Upon entering the electrode region, droplets quickly approach terminal velocity ($\vec{v}_t$), providing an analytical solution to the total lateral displacement of the droplet, assuming that the period at terminal velocity contributes to the majority of deflection (Eq. 3). Here, t_r is the residence time of the droplet in the electrode region, determined by the droplet throughput.

$$\Delta y = \vec{v}_t t_r = \frac{2k\epsilon_{oil}r_d^2V^2}{3\eta_{oil}}t_r \tag{3}$$

3 RESULTS

FEA modeling of dielectrophoretic sorting found three distinct regimes of droplet behavior: no deflection, deflection, and model failure, where the force applied prevents further droplet movement (Figure 1). Sweeping across input voltages showed distinct regions that resulted in successful droplet sorting, given an input droplet diameter or throughput (velocity). However, no single voltage was compatible with all parameter combinations. Preliminary experiments have shown similar regions, where high voltage causes failure by merging adjacent droplets. Varying geometric design parameters will change the total lateral deflection required for sorting (w_i and θ_B) or the resistance ratio of the bifurcation (w_{o1} and w_{o2}), which alter the number of streamlines going

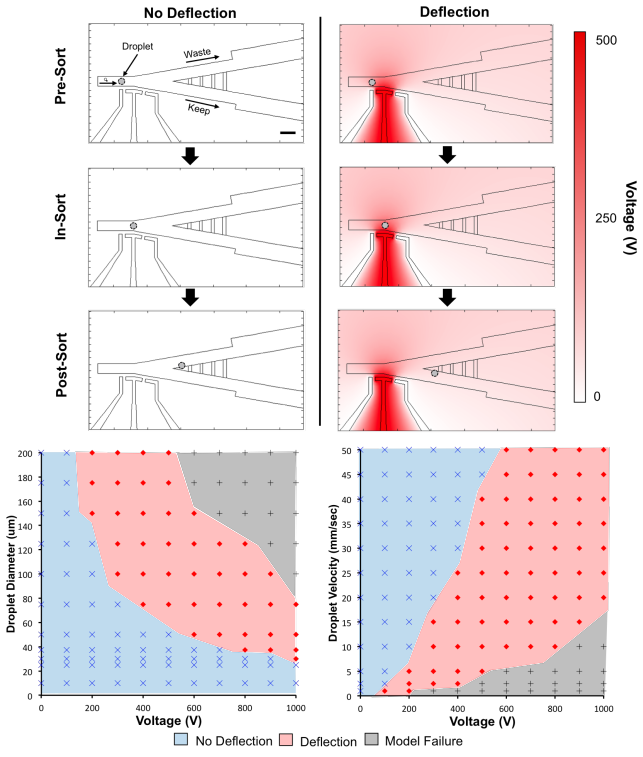
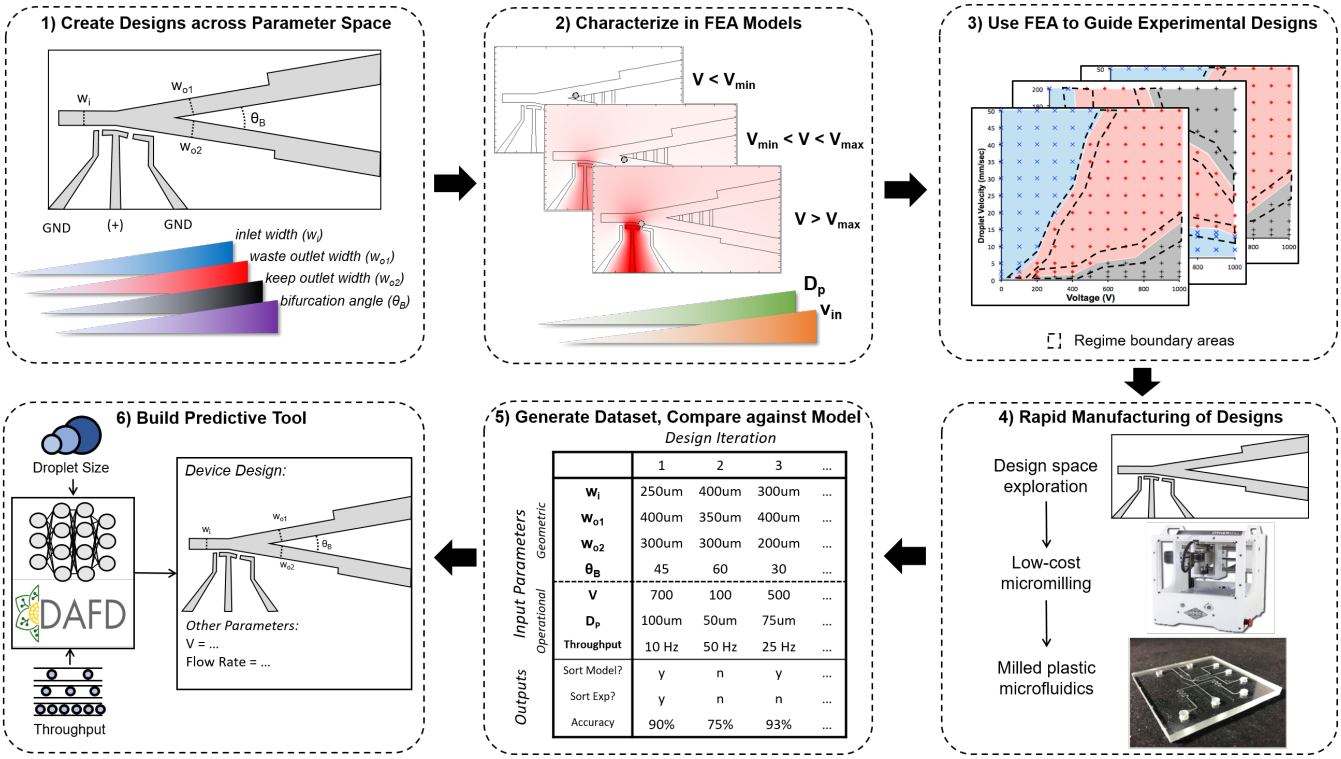


Figure 1: FEA model of droplet microfluidic sorter in different regimes (Top). Effect of voltage, droplet diameter, and velocity on sorter regime (bottom). Scale bar is 250μm.



	1	2	3	...
w_i	250um	400um	300um	...
w_{o1}	400um	350um	400um	...
w_{o2}	300um	300um	200um	...
θ_B	45	60	30	...
V	700	100	500	...
D_p	100um	50um	75um	...
Throughput	10 Hz	50 Hz	25 Hz	...
Sort Model?	y	n	y	...
Sort Exp?	y	n	n	...
Accuracy	90%	75%	93%	...

Figure 2: Workflow of design tool development. Designs covering the parameter space (Box 1) will be characterized by an FEA model (Box 2), reducing the parameter space to regime boundaries (Box 3). These designs will be fabricated with CNC milling (Box 4), compared against the FEA model (Box 5), and used to create a predictive tool based on machine learning (Box 6).

into each channel (Figure 2, Box 1). These initial results highlight the need for a design tool capable of predicting sorter behavior given user specifications.

4 CONCLUSIONS AND FUTURE WORK

This design framework will help guide development of a design automation tool for microfluidic droplet sorting and downstream integration into screening platforms (Figure 2). Further experimental characterization of the design space is needed to assess the FEA model accuracy and validity of simplifying assumptions. Rapid, affordable microfluidic fabrication with CNC milling developed in our group enables collection of the data sets necessary for predictive models, not feasible with standard photolithographic methods [5]. Once developed, this tool will take user-specified droplet size and throughput and return the variable parameters needed for successful, accurate sorting, compatible with the user's detection system of choice. A design automation tool for droplet microfluidic sorting combined with a low-cost fabrication method would enable miniaturization of screening platforms onto the bench-top, increasing accessibility of synthetic biology to non-experts.

Acknowledgements

The author thanks Ali Lashkaripour, Radhakrishna Sanka, and Rabia Yazicigil for their advice throughout the design of this abstract.

REFERENCES

[1] Agresti, J. J., Antipov, E., Abate, A. R., Ahn, K., Rowat, A. C., Baret, J.-C., Marquez, M., Klibanov, A. M., Griffiths, A. D., and Weitz, D. A. Ultrahigh-throughput screening in drop-based microfluidics for directed evolution. *Proceedings of the National Academy of Sciences 107*, 9 (2010), 4004–4009.

[2] Ahn, K., Kerbage, C., Hunt, T. P., Westervelt, R. M., Link, D. R., and Weitz, D. A. Dielectrophoretic manipulation of drops for high-speed microfluidic sorting devices. *Applied Physics Letters 88*, 2 (jan 2006), 024104.

[3] Guo, M. T., Rotem, A., Heyman, J. A., and Weitz, D. A. Droplet microfluidics for high-throughput biological assays. *Lab on a Chip 12*, 12 (may 2012), 2146.

[4] Lashkaripour, A., Rodriguez, C., Ortiz, L., and Densmore, D. Performance tuning of microfluidic flow-focusing droplet generators. *Lab on a Chip 19* (2019), 1041–1053.

[5] Lashkaripour, A., Silva, R., and Densmore, D. Desktop micromilled microfluidics. *Microfluidics and Nanofluidics 22*, 3 (mar 2018), 31.

[6] Lippai, J., Sanka, R., Lashkaripour, A., and Densmore, D. Function-driven, graphical design tool for microfluidic chips: 3duf. International Workshop on Bio-Design Automation (IWBDA).

Detecting Engineering in Single Cells using Tapestri Microfluidics

Aaron Adler[1], Adam Abate[3], Joe Collins[4], Ben Demaree[3], Kevin Keating[4], Xiangpeng Li[3], Thomas Mitchell[1], David Ruff[2], Allison Taggart[1], Shu Wang[2], Daniel Weisgerber[3], Daniel Wyschogrod[1], Fusun Yaman[1], Eric M. Young[4], and Nicholas Roehner[1]

[1]Raytheon BBN Technologies, [2]Mission Bio, [3]University of California San Francisco, [4]Worcester Polytechnic Institute
aaronadler@alum.mit.edu

1 INTRODUCTION

The increased prevalence of engineered organisms increases the importance of detecting such organisms in an automated way. Many types of engineered sequences can be detected by simple string signatures consisting of sequences of nucleotides. However, a major challenge for rare sequence variant detection is the sensitivity of current next-generation sequencing (NGS) technologies. Moreover even when the common bulk population sequencing methods employed by most systems correctly identify rare variants, they lack the ability to resolve whether the variants co-occur within the same cell or originate from different sources within the same heterogeneous sample. Differentiating these cases is key to identify changes that are unusual for a particular cell but might not be unusual for the population as a whole (e.g., a promoter transferred from one species to another).

We have developed a single cell analysis pipeline for yeast as part of our GUARDIAN (Guard for Uncovering Accidental Release, Detecting Intentional Alterations, and Nefariousness) project. The pipeline uses the capabilities of Mission Bio's Tapestri system, a microfluidic device that lyses cells, attaches cell-unique barcodes to DNA fragments, and enables the sequenced DNA to be mapped back to the original cell. By carefully selecting the sequences we look for and then analyzing the co-occurance of the sequences, we can identify whether or not cells have been engineered. This abstract will discuss the selection and design of the engineering signatures for Tapestri, the sample analysis pipeline, and then present our initial results.

2 SIGNATURE DESIGN

The Tapestri platform is shown in Figure 1. Tapestri is a commercial product aimed at detecting cancer cell variants at the single cell level. As part of the GUARDIAN project, we retargeted Tapestri to yeast cells, in particular making changes to the lysis and encapsulation processes to account for the rigid, carbohydrate-based cell wall.

Tapestri uses primer and amplicon panels to detect different DNA sequences within cells. Engineering organisms relies on a set of parts and tools, design rules, and construction methods. We developed a set of 90 primers that target the types of engineering expected in yeast cells; specifically we target detection of plasmids, integration, deletion, and engineering parts. Example targets are shown in Table 1.

Table 1: Engineering Signature Targets

Type	Examples
Plasmid	Bacterial replicons, Yeast replicons, Plasmid DNA
Integration	Integration sites
Deletion	Auxotropic deletions
Engineering	Promoters, Terminators, Fluorescent proteins, Genome editing
Genotyping	Strain identifiers

Once a particular target DNA sequence is selected, Mission Bio has a multi-step panel design workflow. The DNA sequence is compared against reference strains and sites of variation are identified. Depending on the target sequence length, sites of variation, and type of target, primers may be selected at the boundary of or internal to the target sequence. The full set of primers and amplicons are evaluated for specificity, interaction, and GC content among other factors. Following design, the panel is produced and optimized.

3 ANALYSIS PIPELINE

Figure 2 shows the analysis pipeline steps. First samples are prepared and run through the Tapestri. The resulting samples are sequenced, e.g., on a NovaSeq machine. The sequencing data is then trimmed and mapped back to particular DNA signatures using DNA barcodes. The barcodes are then used to map the signatures back to their cells of origin. The last step in the analysis pipeline is to use the combination of signatures to determine whether or not a cell has signs of engineering. The goal of the GUARDIAN project is to determine whether or not a sample contains any engineered cells. The context of signatures (e.g., cell species, presence / absence of other signatures) informs our engineering determinations.

The trimming, mapping, calling, and analysis steps are implemented as Python scripts within a Docker container. We provide a web-based interface to the experimenter to provide metadata about the sample and we record this information in

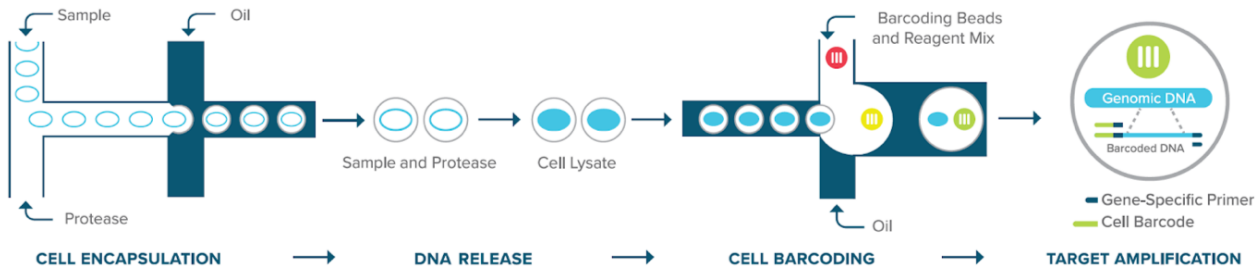


Figure 1: Tapestri processing overview showing how DNA targets are identified in individual cells.

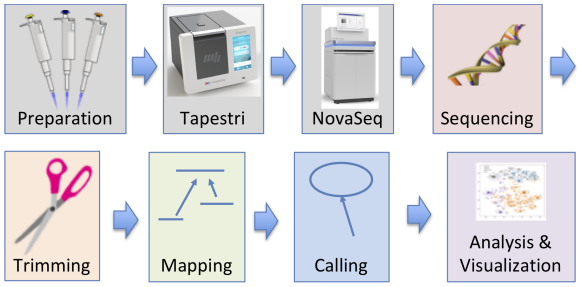


Figure 2: Overview of the GUARDIAN analysis process for identifying whether or not individual cells have been engineered. Signature selection steps are not shown.

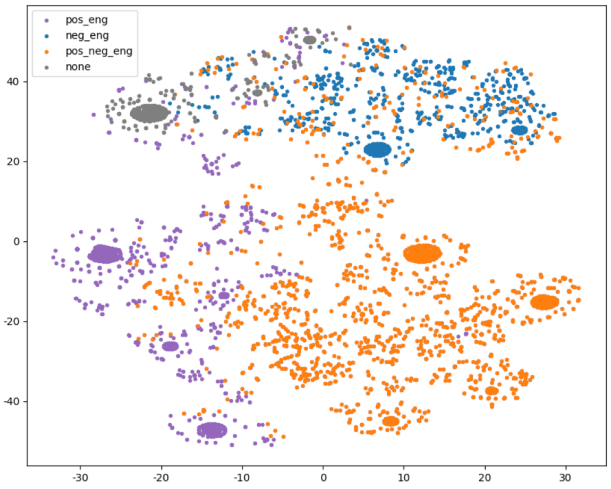


Figure 3: t-SNE plot showing clusters of similar cells labeled based on observed engineering.

a SynBioHub[2] instance. The metadata includes user, date, and sequencing instrument information. The SynBioHub instance also stores information about the amplicons and the target DNA sequences that are referenced by the analysis code. The pipeline leverages open source tools including `bwa` and `samtools`. Each step in the pipeline produces intermediate files, which are labeled with IRIs, and metadata that is stored in SynBioHub. The observed signatures are mapped back to individual cells. The final analysis step is to interpret the presence and absence of particular DNA sequences in the context of the cell type to determine whether or not a cell has been engineered. Each cell is labeled according to the presence or absence of engineering signatures.

Our current approach uses clustering algorithms to identify groups of similar cells. The data can be visualized using t-SNE plots (Figure 3); each dot represents a single cell and the color of the dots indicates whether signatures of engineering were observed (purple), signatures of deletions were observed (blue), signatures of engineering and of deletion were observed (orange), or no signs were seen (gray).

4 DISCUSSION

Our initial work has targeted the CENPK and S288C yeast strains and has demonstrated the ability to identify even low levels of engineering. Our future analysis work will producing detailed reports about the variations seen in individual cells within the larger population. A limitation of this technique is that it can only detect the engineering or other markers that are in a panel. Although the common types of engineering can be robustly detected, this process cannot detect changes that it is not looking for. Our future work will look at ways to address this shortcoming, e.g., single cell whole genome sequencing [1].

ACKNOWLEDGMENTS

This work was supported by the IARPA Finding Engineering-Linked Indicators (FELIX) award HR0011-15-C-0084. This document does not contain technology or technical data controlled under either U.S. International Traffic in Arms Regulation or U.S. Export Administration Regulations.

REFERENCES

[1] Lan, F., Demaree, B., Ahmed, N., and Abate, A. R. Single-cell genome sequencing at ultra-high-throughput with microfluidic droplet barcoding. *Nature biotechnology 35*, 7 (2017), 640.

[2] McLaughlin, J. A., Myers, C. J., Zundel, Z., Misirli, G., Zhang, M., Ofiteru, I. D., Goni-Moreno, A., and Wipat, A. Synbiohub: A standards-enabled design repository for synthetic biology. *ACS Synthetic Biology 7*, 2 (2018), 682–688. PMID: 29316788.

A Logic Programming Language for Computational Nucleic Acid Devices

Carlo Spaccassassi
Microsoft Research
Cambridge, UK

Matthew R. Lakin
University of New Mexico
Albuquerque, NM, USA
mlakin@cs.unm.edu

Andrew Phillips
Microsoft Research
Cambridge, UK
Andrew.Phillips@microsoft.com

1 INTRODUCTION

Computational nucleic acid devices show great potential for enabling a broad range of biotechnology applications, including smart probes for molecular biology research, in vitro assembly of complex compounds, high-precision in vitro disease diagnosis and, ultimately, computational theranostics inside living cells. This diversity of applications is supported by a range of implementation strategies, including nucleic acid strand displacement, localization to substrates, and the use of enzymes with polymerase, nickase, and exonuclease functionality. However, existing computational design tools are unable to account for these strategies in a unified manner.

We present a logic programming language [9] that allows a broad range of computational nucleic acid systems to be designed and analyzed. The language extends standard logic programming with a novel equational theory to express nucleic acid molecular motifs. It automatically identifies matching motifs present in the full system, in order to apply a specified transformation expressed as a logical rule. The language supports the definition of logic predicates, which provide constraints that need to be satisfied in order for a given rule to be applied.

Our language can encode the semantics of nucleic strand displacement systems with complex topologies, previous extensions to the Visual DSD language [6], as well as new extensions including the encoding of kinetic rate hypotheses, together with computation performed by a broad range of enzymes. More importantly, our approach is extensible in that new nucleic acid implementation strategies can be encoded simply by defining new logic predicates. Thus, our approach lays the foundation for a unifying framework for the design of computational nucleic acid devices.

2 LANGUAGE DESIGN

Our approach relies on the *domain level abstraction*, which has been used to successfully design and engineer a broad range of computation DNA systems. This abstraction assumes that sequences have been carefully designed to allow consecutive nucleotides to be considered as distinct, non-interfering domains, and thereby provides a high-level conceptual framework for modelling and designing the behavior of nucleic acid computational devices.

The syntax of our language (Figure 1A) is formally defined in the style of process calculi and combines a syntax for nucleic acid strands, which extends previous work on strand graphs [8], with a syntax for logic programs, based on Prolog. Compiling a model written in our logic programming language uses a custom inference system based on *SLDNF resolution*, which is widely used in logic programming languages such as Prolog. We simplified the treatment of negative predicates to avoid a consistency problem with standard SLDNF known as *floundering*, modified the standard resolution strategy to a breadth-first search (which is guaranteed to find all possible solutions), and, most importantly, extended the standard unification algorithm with a novel **equational theory of nucleic acid strands**. Our equational theory defines *contexts* and *patterns* that allow the identification and sound manipulation of general nucleic acid motifs.

3 PROTOTYPE IMPLEMENTATION

A prototype implementation of our logic programming language has been integrated in Visual DSD and is freely accessible at `http://classicdsd.azurewebsites.net`, together with a collection of built-in *Logic Programming* examples. This integration means that all of the existing simulation and analysis methods present in Visual DSD are also supported by our logic programming language, providing a strong motivation for implementing our logic programming language as an extension to Visual DSD, rather than as a Prolog library. Our logic programming language can encode previous extensions to the Visual DSD language [4, 5, 8, 10], as well as new extensions including enzyme interactions [7] and the encoding of kinetic rate hypotheses [1], neither of which were previously supported in Visual DSD. More importantly, our approach is extensible in that new nucleic acid implementation strategies can be encoded simply by defining new logic predicates. Thus, our work provides a framework to model the largest set of nucleic acid information processing systems to date (Figure 1B). For a selection of examples that are also supported by previous versions of Visual DSD [5], our logic programming language has an average perfomance penalty of approximately 25% in terms of compilation time. This is not unexpected given the generality of the language and we are working to optimize the system further.

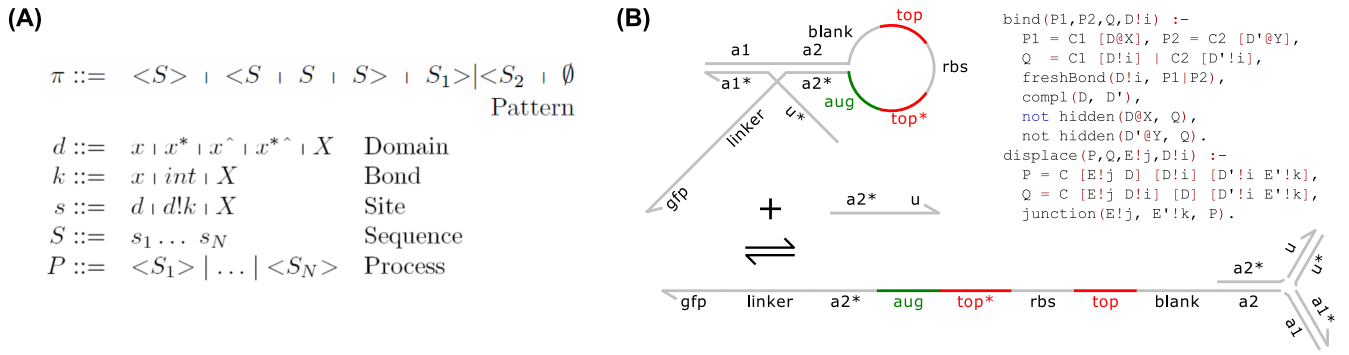


Figure 1: (A) BNF syntax for patterns and processes. (B) Part of a chemical reaction network for a ribocomputing AND gate [3], and corresponding logic program code to express domain binding and strand displacement. These rule sets are extensible: additional rules could be added to encode translation of GFP, for example.

4 DISCUSSION

We envisage a number of usage scenarios for our logic programming language. In the most common scenario, we anticipate that the basic rules for nucleic acid strand displacement will be included by default, and the user will add new enzyme rules or kinetic hypotheses depending on the particular implementation strategy. More broadly, we anticipate two different classes of users: those who write predicates that define particular nucleic acid implementation strategies, and those who select from a set of existing predicates to model their systems of interest. This will allow scientists not familiar with logic programming to still take advantage of the enhanced customisation that our approach enables.

Our logic programming approach invites comparison with Kappa [2], which is a rule-based modelling language that captures the interactions between agents via named sites. While Kappa rules define patterns that can be matched to a system, our approach further allows the definition of arbitrary logical predicates to be associated with a rule, allowing increased generality. For example, our approach allows a single binding rule to be applied for all domains that are complementary, whereas Kappa would require a separate binding rule to be written for each specific domain. Furthermore, our approach allows complex topologies to be expressed using predicates, such as whether a domain is hidden inside a hairpin, or is part of a junction of arbitrary size. An interesting goal would be to encode Kappa in our framework, so one could extend Kappa with arbitrary logical predicates.

As a practical matter, a useful future direction would be the inclusion of a formal *module system* whereby particular sets of predicates can be defined as self-contained modules and easily loaded into a model with a single command. This would enable the creation of a *standard library* of commonly used predicate sets for well-known experimental frameworks such as DNA strand displacement or the PEN-DNA toolbox.

5 ACKNOWLEDGMENTS

Boyan Yordanov, Filippo Polo, and Rasmus Petersen implemented visualizations and integrations with Visual DSD. This material is based upon work supported by the NSF under Grant Nos. 1518861, 1525553, and 1814906 to M.R.L.

REFERENCES

[1] Chen, Y.-J., Dalchau, N., Srinivas, N., Phillips, A., Cardelli, L., Soloveichik, D., and Seelig, G. Programmable chemical controllers made from DNA. *Nature Nanotechnology 8* (2013), 755–762.

[2] Danos, V., Feret, J., Fontana, W., Harmer, R., and Krivine, J. Rule-based modelling of cellular signalling. In *CONCUR 2007* (2007), L. Caires and V. T. Vasconcelos, Eds., vol. 4703 of *Lecture Notes in Compuer Science*, Springer-Verlag, pp. 17–41.

[3] Green, A. A., Jongmin Kim and, D. M., Silver, P. A., Collins, J. J., and Yin, P. Complex cellular logic computation using ribocomputing devices. *Nature 548* (2017), 117–121.

[4] Lakin, M. R., Petersen, R., Gray, K. E., and Phillips, A. Abstract modelling of tethered DNA circuits. In *Proceedings of the 20th International Conference on DNA Computing and Molecular Programming* (2014), S. Murata and S. Kobayashi, Eds., vol. 8727 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 132–147.

[5] Lakin, M. R., Youssef, S., Cardelli, L., and Phillips, A. Abstractions for DNA circuit design. *Journal of the Royal Society Interface 9*, 68 (2012), 460–486.

[6] Lakin, M. R., Youssef, S., Polo, F., Emmott, S., and Phillips, A. Visual DSD: a design and analysis tool for DNA strand displacement systems. *Bioinformatics 27*, 22 (2011), 3211–3213.

[7] Montagne, K., Plasson, R., Sakai, Y., Fujii, T., and Rondelez, Y. Programming an *in vitro* DNA oscillator using a molecular networking strategy. *Molecular Systems Biology 7* (2011), 466.

[8] Petersen, R. L., Lakin, M. R., and Phillips, A. A strand graph semantics for DNA-based computation. *Theoretical Computer Science 632* (2016), 43–73.

[9] Spaccassassi, C., Lakin, M. R., and Phillips, A. A logic programming language for computational nucleic acid devices. *ACS Synthetic Biology* (2018). Published online ahead of print.

[10] Yordanov, B., Kim, J., Petersen, R. L., Shudy, A., Kulkarni, V. V., and Phillips, A. Computational design of nucleic acid feedback control circuits. *ACS Synthetic Biology 3*, 8 (2014), 600–616.

CellScanner: a software for extracting physical features from individual cells

Sebastián Antón[1], **Marco Clavero**[1], **Martín Gutiérrez**[1*]
[1]Escuela de Informática y Telecomunicaciones - Universidad Diego Portales
Santiago, Chile
{sebastian.anton,marco.clavero,martin.gutierrez}@mail.udp.cl

1 INTRODUCTION

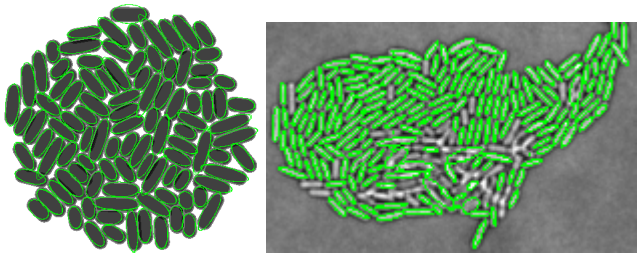
Characterization and classification of microbial organisms prove to be interesting and important tasks, as much is still unknown about their features and available data is not precise. Often this is a problem, as these estimates are not accurate and cannot be used as a reliable source for simulating organism behavior. This problem is aggravated by the fact that simulation designers are unaware of the variables to take into account when producing their solution. Automation has proven to be a great aid in eliminating human error and delivering standardized solutions. As such, automating the retrieval of individual cells from graphical material, such as pictures or videos, allows for the generation of massive amounts of data. In turn, processing these data would improve the estimates for currently existing cell features, acting as better parameters within a simulation environment.

2 THE PROBLEM

Extraction of cell features is important. It is a tedious process in which observation, retrieval and analysis are required. This takes up research time from the scientist. Applications stemming from the retrieved data include diagnostics, simulation and characterization. The automation of feature extraction would offer a platform for data basic analysis and storage. It should release the scientist from the task, using input graphical material such as pictures and/or videos.

3 PROPOSED SOLUTION

The goal of this work is to be able to identify a large amount of individual cells and extract important features in an automated manner. Features such as length, width and orientation angle are of interest. Previous work has been done on this topic [5, 7, 10], however they provide insufficient information to extract for dynamical behavior of cells. Pictures and/or videos of growing cell colonies should serve as input to the tool. In our proof-of-concept, 2D `gro` [3, 4] simulations pictures and videos were used, and also pictures in the literature showing cell colonies [2, 9]. Computer vision techniques are at the core of this solution. Specifically, the Python OpenCV library [8] was used to process the images/videos. Cells must be identified and segmented as a first step. However, preprocessing the input file first is necessary to better prepare the object borders: operations such as erosion and dilation are used. Once the file is preprocessed, a segmentation algorithm is applied (Canny Edge [1]) to identify the contours of each cell. Once this step is completed, noise is filtered through approximation to a geometric shape (an ellipse in the proof-of-concept). The parameters (length, width, orientation angle) are then evaluated and stored for further analysis. An example is shown in Figure 1. This process is followed by any kind of input, but describes the full process for pictures. However, if the input file is a video, further operations must be performed. These operations are concerned with the location tracking of a cell object. This is done by searching in a spatial neighborhood of the projected location of the object (optical flow algorithm [6]), and selecting the new location closest to it. Identification of the key events (cellular division, for instance) is also tracked by constantly observing cell length (Figure 2). Therefore, cell division time can be recorded. Figure 3 shows the frames before and after division.



(a) Simulation image **(b) Microscopy image [2]**

Figure 1: Individual cells are identified and approximated by an ellipse for data extraction.

4 RESULTS

Both image and video tests were run. For the simulation image in Figure 1, a segmented bacterial count of 1578 was reached out of 1731 possible bacteria (91.16% accuracy), while in the microscopy image 225 cells out of 249 (90.36% accuracy) were segmented. False positives may occur if an empty

*All authors contributed equally to this research. This research was funded by Semilla Project MGUTIERREZ - UDP.

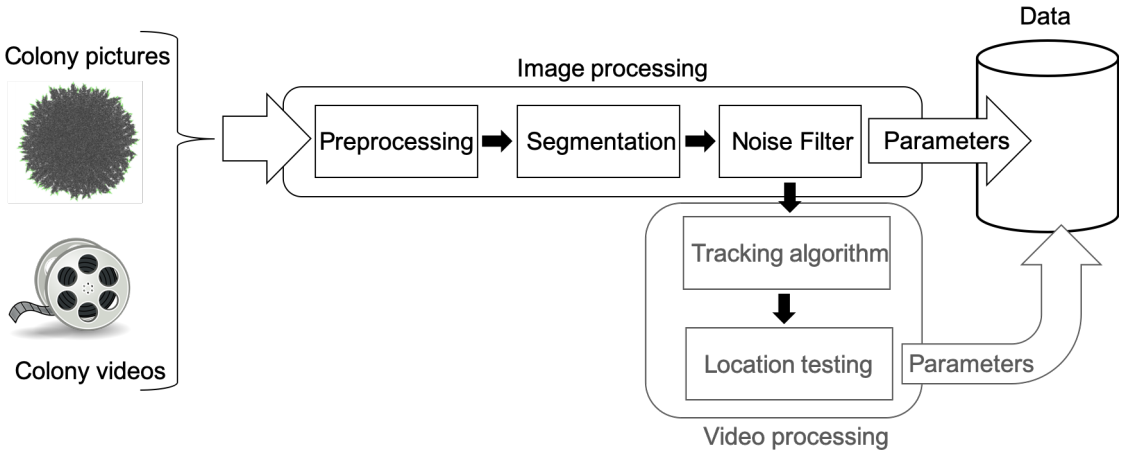
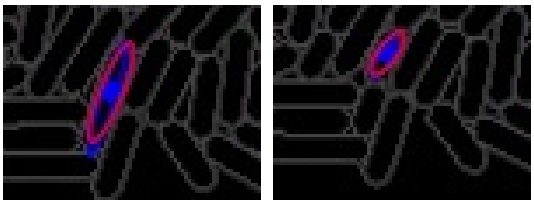


Figure 2: Parameter extraction process. The graphical material is processed by applying several operations on the files. Individual cells are identified by segmenting the image/video and data are extracted from each cell. In case a video is being processed, then an individual cell is tracked and tested for its location to obtain dynamical behavior parameters.



(a) Tracked cell ready to divide **(b) Tracked cell after division**

Figure 3: Sample frames of a gro simulation video in which single cell tracking is occuring. This tracking can be extended to all bacteria in the colony.

space with a similar size to a cell is present. Object filters are applied onto the images to reduce the amount of detected false positives. Width, maximum and minimum length, location (x and y coordinates, in pixels) and the orientation angle were the features extracted for each bacterium. Minimum length/maximum length and width/maximum length ratios of 1:3 and 1:4 (simulated), and 2:3 and 1:4 (microscopy) were found, respectively. For videos, cells are tracked and consecutive division events for each tracked cell are recorded.

5 CONCLUSIONS AND FUTURE WORK

Our method for identification of individual cells in pictures and their tracking in videos and later extracting data has initially proven successful. Accuracy of the method is being improved, and we are also working to include more features (such as shape, 3D body approximation or division dynamics). Other Machine Learning techniques are being added to the method. Many tests on other species remain to be run. The overall goal is to automate the characterization and representation of cells for simulation through data.

REFERENCES

[1] Canny, J. A computational approach to edge detection. In *Readings in Computer Vision*. Elsevier, 1987, pp. 184–203.

[2] de Jong, I. G., Beilharz, K., Kuipers, O. P., and Veening, J.-W. Live cell imaging of bacillus subtilis and streptococcus pneumoniae using automated time-lapse microscopy. *JoVE (Journal of Visualized Experiments)*, 53 (2011), e3145.

[3] Gutierrez, M., Gregorio-Godoy, P., Perez del Pulgar, G., Muñoz, L. E., Sáez, S., and Rodríguez-Patón, A. A new improved and extended version of the multicell bacterial simulator gro. *ACS synthetic biology 6*, 8 (2017), 1496–1508.

[4] Jang, S. S., Oishi, K. T., Egbert, R. G., and Klavins, E. Specification and simulation of synthetic multicelled behaviors. *ACS Synthetic Biology 1*, 8 (2012), 365–374.

[5] Li, K., and Kanade, T. Nonnegative mixed-norm preconditioning for microscopy image segmentation. In *International Conference on Information Processing in Medical Imaging* (2009), Springer, pp. 362–373.

[6] Lucas, B. D., and Kanade, T. An iterative image registration technique with an application to stereo vision.

[7] McQuin, C., Goodman, A., Chernyshev, V., Kamentsky, L., Cimini, B. A., Karhohs, K. W., Doan, M., Ding, L., Rafelski, S. M., Thirstrup, D., et al. Cellprofiler 3.0: Next-generation image processing for biology. *PLoS biology 16*, 7 (2018), e2005970.

[8] Minichino, J., and Howse, J. *Learning OpenCV 3 Computer Vision with Python*. Packt Publishing Ltd, 2015.

[9] Su, P.-T., Liao, C.-T., Roan, J.-R., Wang, S.-H., Chiou, A., and Syu, W.-J. Bacterial colony from two-dimensional division to three-dimensional development. *PloS One 7*, 11 (2012), e48098.

[10] Wang, Q., Niemi, J., Tan, C.-M., You, L., and West, M. Image segmentation and dynamic lineage analysis in single-cell fluorescence microscopy. *Cytometry Part A: The Journal of the International Society for Advancement of Cytometry 77*, 1 (2010), 101–110.

OptBioDes: optimal design for the synbio toolchain

Pablo Carbonell, Rainer Breitling, Jean-Loup Faulon, The SYNBIOCHEM Team
pablo.carbonell@manchester.ac.uk
SYNBIOCHEM, Manchester Institute of Biotechnology, University of Manchester
Manchester, UK

1 INTRODUCTION

The once future vision of green, sustainable manufacturing of valuable chemicals for industrial sectors such as materials, drugs, cosmetics, and food is suddenly becoming the biomanufacturing of today by embracing bio-based solutions using natural and renewable energy sources. The bioproduction of chemicals in its transition from an emerging applied science to a truly industrial biomanufacturing technology is requiring an accelerated reduction of delivery times from concept to development to scale-up. In order to achieve its full potential, the industrial biomanufacturing pipeline demands strategies for rapid prototyping [5], streamlined scaling from microtiter plates to commercial bioreactors, and robust upstream and downstream processing [11]. To that end, top-down bioprocess engineering has often relied on optimal experimental design approaches [4, 8], delivering in that way an approach to a) make lean use of resources; b) facilitate agile updates in the pipeline; c) perform an efficient identification of main effects in the factorial design; and d) enable modeling and provide predictive ability. This type of design approach has been widely adopted in sectors such as defense, manufacturing, or pharma, where proprietary design software have been generally the option of choice.

In contrast, the development of bottom-up approaches for engineering biology [6] has followed a different path because of its collaborative environment where constructs, strains and protocols are shared across the labs. Adoption of the optimal design of experiments has remained slow [2, 3], arguably hampered by the lack of open community standards. To contribute to the genuine open spirit of the synthetic biology community, we want to provide here `OptBioDes`, an open-source community-based optimal design tool that is tailored to the synbio-based chemical production of biomanufacturing pipelines and provides full design diagnostics capabilities. Following the model of cloud labs, `OptBioDes` lives in the cloud, and it is also available as a Galaxy node for workflow development [1].

2 MATERIALS AND METHODS

The problem under study consists of the generation of optimal combinatorial libraries of plasmids that are built in order to efficiently express a metabolic pathway for chemical production under different conditions and in a given chassis organism. Factors that typically enter into the design choice are plasmid copy number of the vector backbone (ori), resistance cassette (res), promoter (pro), open reading frame (orf) of the gene variants (including RBS libraries) associated with each of the n steps, terminator (ter) and gene arrangement. The construct has the following constraints:

(1) Layout: $\mathrm{ori}+\mathrm{res}+\mathrm{pro}_1+\mathrm{orf}_1+\sum_{i=2}^{n}([\mathrm{ter}_{i-1}+\mathrm{pro}_i]+ +\mathrm{orf}_i)+\mathrm{ter}_n$,
(2) Arrangement: orfs correspond to gene variants of each step, which may either appear at predefined positions or in free arrangement,

where $[\mathrm{ter}_{i-1}+\mathrm{pro}_i]$ might be present or absent, depending on the design specifications. In our approach, we assume no prior knowledge about the effects of the genetic parts on the desired phenotype of the engineered strains. Therefore, factors are coded using one-hot encoding in an hypercube with values -1 and 1 for the $l-1$ levels, corresponding to their number of degrees of freedom and expressed as normalized vectors in their $l-1$ principal directions.

Our optimal experimental design is based on a logistic regression analysis with an assumed model for the response $\mathbf{Y}: \mathbf{Y}=\beta\mathbf{X}+\alpha$, where $\mathbf{X}$ is defined as the model matrix, i.e., a row for each experimental run and a column for each term in the model. We consider D-optimal designs, where the design is evaluated based on its D-efficiency, which compares the design with an orthogonal design expressed as a percentage:

$$D_{eff} = 100\left(\frac{1}{n}|\mathbf{X}^T\mathbf{X}|^{1/p}\right), \quad (1)$$

where p is the number of independent variables or degrees of freedom, $\mathbf{X}$ is the model matrix, and n the library size.

Orthogonal designs are defined as follows [10]:

- the entries in the model matrix $\mathbf{X}$ are either -1 or 1,
- the columns are orthogonal $c_i^T c_j = 0$ for $i \neq j$,
- $c_i^T c_i = n$, where n is the number of experiments,
- the sum of the absolute value of the columns is n.

`OptBioDes` is a `Python` library for synbio optimal design whose source code is available at https://github.com/pablocarb/doebase. In order to generate a D-optimal experimental design for the given factors, constraints and underlying model, we used an implementation of the `Coordinate Exchange` algorithm [7]. A cloud-based application for `OptBioDes` can be accessed through REST web services at http://optbiodes.synbiochem.co.uk/REST. For the returned designs, the tool provides diagnostics such as D-efficiency (Equation 1), power

analysis and relative prediction variance [7], allowing the optimal selection of library size. In order to integrate OptBioDes into pathway design workflows, a Galaxy tool is available as part of the synbiodesign repository of the Galaxy Tool Shed.

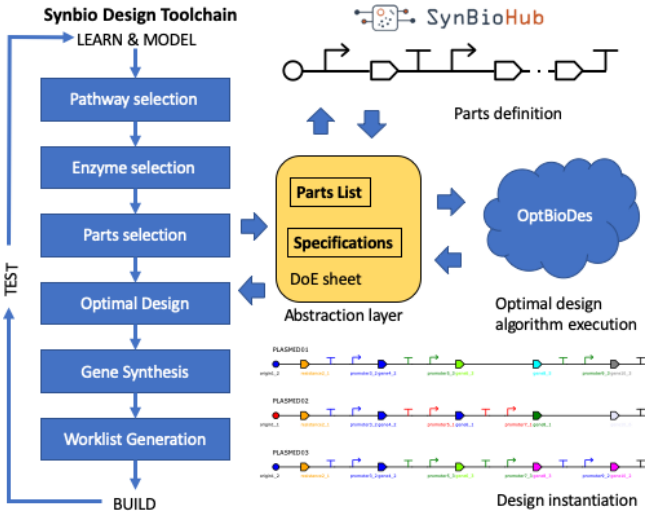


Figure 1: OptBioDes in the synbio design–build–test–learn toolchain. Modeling and learn guide pathway, enzyme and genetic parts selection in order to generate a DoE specification sheet, which acts as an abstraction layer with respect to the parts definition. The sheet is submitted to the optimizer and the result is an instantiation of the experimental design according to the specifications.

3 RESULTS

The key strength of OptBioDes is its ability to establish an optimal automated design within the synbio design toolchain. As shown in Figure 1, the tool seamlessly integrates into the design–build–test–learn pipeline and can be used as part of a full workflow using Galaxy or other automation environments. Starting from the target chemical, information is passed through parts selection tools. Resulting selection is then compiled in a DoE sheet (see GitHub site for details), where genetic part information can be referenced from a central repository like SynBioHub [9]. Following the cloud biomanufacturing paradigm, once the optimal combinatorial designs have been selected, the automated build and test stages proceed through robotic platforms allowing full sample traceability so that experimental results are fed back into the learn and design engines. Statistical analyses are then applied in order to infer an ensemble of models for the design factor–response relationships under either mechanistic hypothesis, i.e., kinetic models, or using model-free approaches, i.e., machine learning. Models considering more complex interactions between factors will be supported in the future. The resulting analyses provide a new set of design rules for the next experimental round.

For instance, our optimal design approach was employed in order to design highly-compressed 16-member libraries for 4-gene producer pathways for flavonoids and alkaloids production in *E. coli* [5]. For both classes of compounds the application of the first iteration of rapid prototyping either led to the identification of high producer strains or provided enough information in order to narrow down main effects for a rapid second iteration focused on high producers. Such compelling results show that this fully automated optimal design node plug-in for the synbio toolchain complies with the flexibility requirements of agile design and is therefore expected to become widely adopted in future biomanufacturing systems.

4 ACKNOWLEDGEMENTS

This project has received funding from the UK BBSRC under Grant BB/M017702/1, "Centre for Synthetic Biology of Fine and Specialty Chemicals" and from the European Union's Horizon 2020 research and innovation programme under Grant Agreements No. 814408 "ShikiFactory100", and No. 730976 "Industrial Biotechnology Innovator and Synthetic Biology Accelerator" (IBISBA).

REFERENCES

[1] Afgan, E., Baker, D., van den Beek, M., Blankenberg, D., et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Res. 44*, W1 (2016), W3–W10.

[2] Bhatia, S., Smanski, M., Voigt, C. A., and Densmore, D. Genetic design via combinatorial constraint specification. *ACS Synth. Biol.*, 11 (2017), 2130–2135.

[3] Braniff, N., and Ingalls, B. New opportunities for optimal design of dynamic experiments in systems and synthetic biology. *Curr. Opin. Sys. Biol. 9* (2018), 42–48.

[4] Campbell, K., Xia, J., and Nielsen, J. The impact of Systems Biology on Bioprocessing. *Trends in Biotechnol.*, 12 (2017), 1156–1168.

[5] Carbonell, P., Jervis, A. J., Robinson, C. J., Yan, C., et al. An automated Design-Build-Test-Learn pipeline for enhanced microbial production of fine chemicals. *Comm. Biol. 1*, 1 (2018), 66.

[6] Choi, K. R., Jang, W. D., Yang, D., Cho, J. S., et al. Systems Metabolic Engineering strategies: integrating Systems and Synthetic Biology with Metabolic Engineering. *Trends in Biotechnol.* (2019).

[7] Goos, P., and Jones, B. *Optimal design of experiments : a case study approach.* Wiley, 2011.

[8] Kumar, V., Bhalla, A., and Rathore, A. S. Design of experiments applications in bioprocessing: Concepts and approach. *Biotechnol. Prog. 30*, 1 (2014), 86–99.

[9] McLaughlin, J. A., Myers, C. J., Zundel, Z., Misirli, G., et al. SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology. *ACS Synth. Biol. 7*, 2 (2018), 682–688.

[10] Olive, D. J. Orthogonal Designs. In *Linear Regression*. Springer International Publishing, Cham, 2017, pp. 245–282.

[11] Wehrs, M., Tanjore, D., Eng, T., Lievense, J., Pray, T. R., and Mukhopadhyay, A. Engineering robust production microbes for large-scale cultivation. *Trends in Microbiol.*, 6 (2019), 524–537.

Integration of Performance Metrics into Microfluidic Design Automation

Radhakrishna Sanka
sanka@bu.edu
Boston University

Douglas Densmore
dougd@bu.edu
Boston University

1 INTRODUCTION

The design of microfluidic devices has the potential to accelerate the discovery and the exploration of engineering paradigms [5]; however, the non-standardized technology stacks coupled with the tedious manual design processes have prevented the complete integration of microfluidic technologies into academic and research settings [10]. While a majority of the research in Microfluidic Design Automation (MDA) tools has been focused on physical design automation and the generation of robust architectures, recent works by Grimmer et al. have explored the possibility of integrating device sizing and architecture generation into the design workflows [2, 3]. However, a significant deficit in these methods is their complete reliance on analytical models which are based on simplifying assumptions that often do not represent the experimentally observed behavior into their formulation. In order to bridge the gap between abstractions provided by MDA tools and the generation of realistic devices, we outline a design flow that enables the integration of performance metrics into the MDA design flows enabled by 3DμF [9], Fluigi [4] and DAFD [7].

2 PERFORMANCE CHARACTERIZATION

Because of the presence of complex surface interactions and fluid dynamic phenomenon, it is very difficult to characterize the behavior of microfluidic devices. By taking advantage of rapid prototyping techniques [8], DAFD employs machine learning algorithms to characterize and predict microfluidic behavior [6] over a large design space.

In order to use DAFD, the user first needs to generate a dataset that sufficiently characterizes a component by varying the flow conditions and its geometric parameters. After training the performance models within DAFD using the data from characterization experiments, the user would be able to query DAFD for a desired performance alongside the geometric parameters and their respective flow conditions.

3 SPECIFICATION OF PERFORMANCE METRICS

Liquid Flow Relations (LFR) is a hardware description language (HDL) that borrows syntax and concepts from Verilog [1] for describing devices that perform liquid manipulations. LFR allows the designer to abstract the technologies that are used for performing the fluid manipulations and describe the entire device's behavior in terms of how the different fluids that would be injected onto the device would be distributed and controlled.

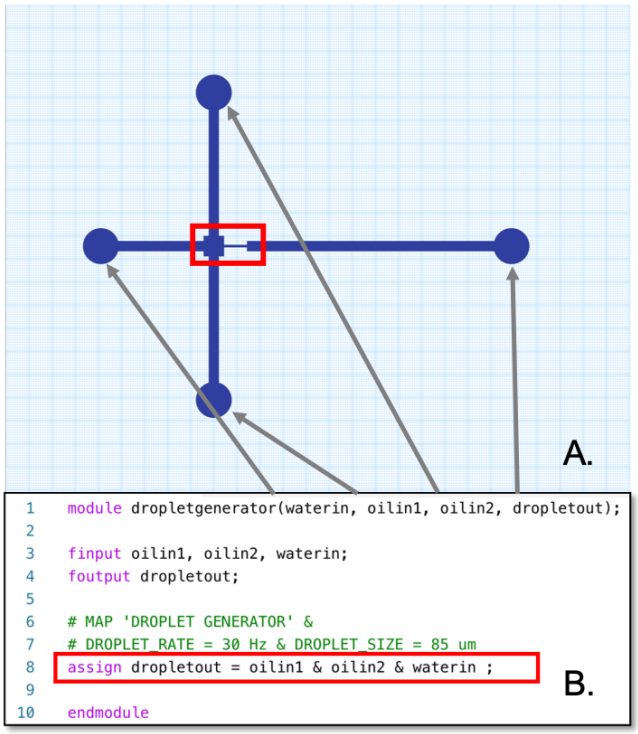


Figure 1: Specifying Droplet Generator performance using LFR - A. Shows a representative design of a device that contains a droplet generator. B. Gives the LFR (Liquid Flow Relations) description of the device shown in A. Lines 6-7 how the user can define custom performance constraints while Lines 1 and 8 show how different elements of the device description allow for the synthesis of the design architecture.

Figure 1 shows an example of an LFR specification that is used to describe a device used to generate droplets. Lines 1 and 10 give a top level description where the user explicitly specifies the inputs and outputs. In order to assign the performance specification, the user needs to annotate the *assign* statement (Line 8) to help LFR synthesize a *DROPLET GENERATOR* with the target performance metrics (Lines 6 and 7).

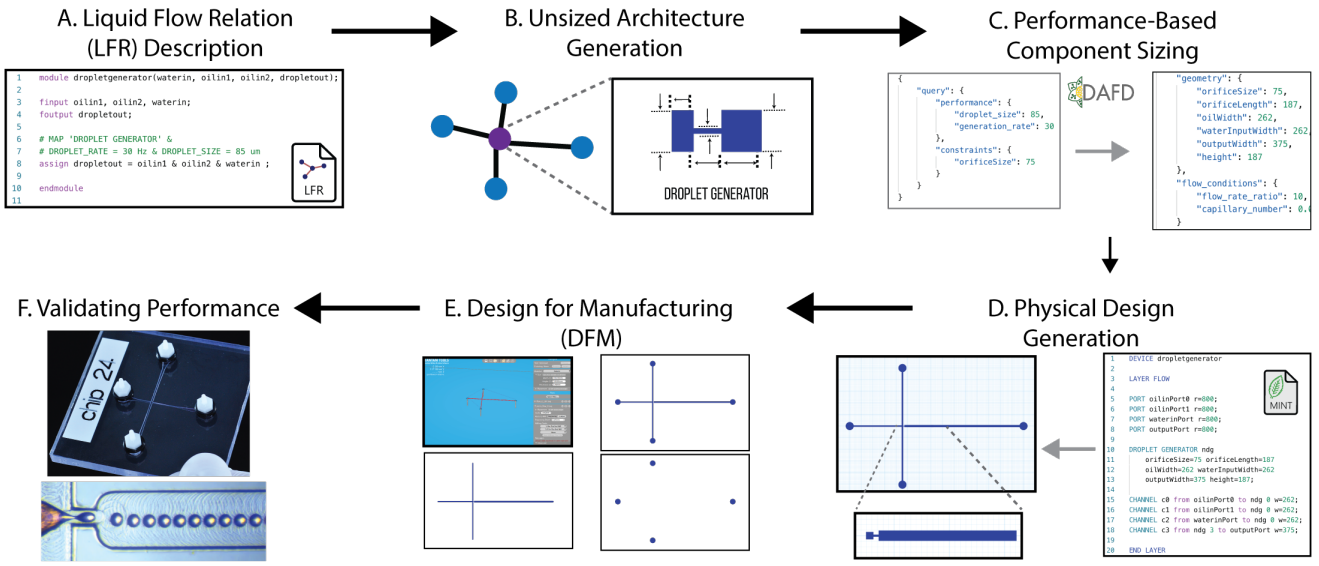


Figure 2: Steps A-F describe how a text based Liquid Flow Relation (LFR) file is converted into a device that matched the performance specification described in the LFR file. The LFR compiler reads the description and creates an unsized architecture (incorrect dimensions). The compiler then queries DAFD with the performance specifications and retrieves component parameters to generate the final design description that is used by Fluigi[4] to generate the layout of the device and the subsequently generate the DFM output that are optimized for different manufacturing processes to be manufactured and validated.

4 ARCHITECTURE SYNTHESIS

In order to generate the microfluidic architecture from LFR specification, the compiler first constructs a *Fluid Interaction Graph* that captures the interactions and behaviors between the various fluid inputs. The compiler then processes the graph to evaluate the logical expressions that are associated with the control inputs and finally maps microfluidic technologies to generate a netlist G that describes the architecture of the device. Where $G \in (V, E)$ is a graph where V is collection of components and E is a collection of connections that link various components.

Once the initial netlist is generated, the LFR compiler sequentially goes through each of the performance specifications associated with netlist and queries DAFD for the geometric parameters that are necessary for individual components to meet their respective performance targets.

5 CONCLUSION AND FUTURE WORK

The integration of performance metrics into the larger MDA workflows (Architecture Synthesis and Physical Design Automation) has the potential to reduce the amount of microfluidic expertise needed to takes ideas seen in literature and apply them in different applications. By expanding the datasets utilized by DAFD to include different microfluidic design primitives, we can extend the capabilities and the efficacy of MDA tools and allow for new synergies between researchers working in microfluidics research and design automation.

REFERENCES

[1] IEEE 1800-2017 - IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language.

[2] Grimmer, A., Haselmayr, W., Springer, A., and Wille, R. A discrete model for Networked Labs-on-Chips: Linking the physical world to design automation. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)* (June 2017), pp. 1–6.

[3] Grimmer, A., Haselmayr, W., and Wille, R. Automated Dimensioning of Networked Labs-on-Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018), 1–1.

[4] Huang, H. *Fluigi: An end-to-end Software Workflow for Microfluidic Design.* PhD thesis, Boston University, 2015.

[5] Huang, H., and Densmore, D. Integration of microfluidics into the synthetic biology design flow. *Lab on a Chip 14*, 18 (Aug. 2014), 3459–3474.

[6] Lashkaripour, A., Rodriguez, C., Ortiz, L., and Densmore, D. Performance tuning of microfluidic flow-focusing droplet generators. *Lab on a Chip 19*, 6 (Mar. 2019), 1041–1053.

[7] Lashkaripour, A., Sanka, R., Lippai, J., and Densmore, D. Design automation based on fluid dynamics, 2017.

[8] Lashkaripour, A., Silva, R., and Densmore, D. Desktop micromilled microfluidics. *Microfluidics and Nanofluidics 22*, 3 (Mar. 2018), 31.

[9] Sanka, R., Lippai, J., Samarasekera, D., Nemsick, S., and Densmore, D. 3dμf - interactive design environment for continuous flow microfluidic devices. *Scientific Reports (In submission)* (2019).

[10] Whitesides, G. M. The origins and the future of microfluidics. *Nature 442*, 7101 (July 2006), 368–373.

Modular Microfluidic Design Automation Using Machine Learning

Ali Lashkaripour
Biomedical Engineering Department, Boston University
lashkari@bu.edu

Christopher Rodriguez
Department of Cyber Engineering, Louisiana Tech University
cwr023@latech.edu

Noushin Mehdipour
Division of System Engineering, Boston University
noushinm@bu.edu

David McIntyre
Biomedical Engineering Department, Boston University
dpmc@bu.edu

Douglas Densmore*
Department of Electrical and Computer Engineering, Boston University
dougd@bu.edu

1 INTRODUCTION

Microfluidics is the science of handling liquids inside sub-millimeter microchannels at nano-liter and pico-liter scales. This volume reduction enables increased resolution, sensitivity, and throughput, while, reducing the reagent cost significantly [1, 11]. These advantages make microfluidic devices to be ideal substitutes for bench-top and robotic liquid handling in numerous life science applications, specifically, synthetic biology where there is a need for low-cost, automated, and high-throughput platforms [3]. Despite the need, implementing microfluidic platforms in the life science research work-flow is an exception rather than being the norm [8, 10]. This can be attributed to the high cost of fabricating microfluidic devices and a lack of microfluidic design automation tools that can design a microfluidic geometry based on the desired performance [6]. As a result, designing a microfluidic device that delivers an expected performance is an iterative, time-consuming, and costly process [2]. To address this, we previously described a low-cost and accessible micro-milling technique to fabricate microfluidic devices in less than an hour while costing less than $10 [7]. However, still designing a microfluidic device that performs as expected is an iterative and in-efficient process. Therefore, microfluidic design automation tools that are able to design a microfluidic geometry and provide the necessary flow conditions and fluid properties that would deliver a user-specified performance is with great importance. We propose a modular design automation tool, called DAFD, that is able to design a microfluidic device based on user-specified performance and constraints. DAFD uses machine learning to generate accurate predictive models, and then exploits these models to provide a design automation platform. DAFD can be implemented in many microfluidic applications such as droplet generation [4], high-throughput sorting, and micro-mixing.

2 A MODULAR DESIGN AUTOMATION TOOL

DAFD can be divided into two sections. First a forward predictive model that is built by choosing the most accurate algorithm from the built-in machine learning algorithms. Once an accurate model is identified by the user, this model is able to predict the performance of the systems if the inputs are given. Second, the reverse model (i.e., design automation tool) that allows the user to specify the desired performance and get the required inputs in return, as shown in Fig. 1.

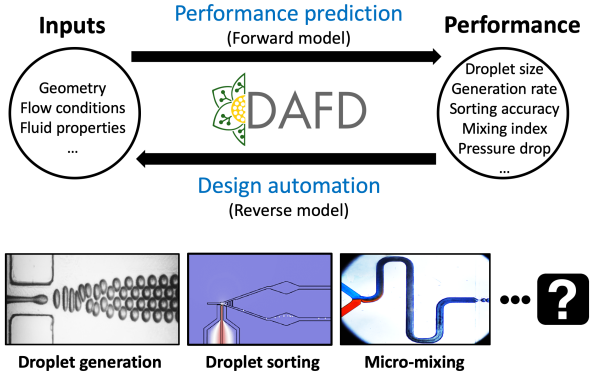


Figure 1: DAFD, relies on a forward predictive model that relates the inputs of the system to its performance. Using this predictive model the design automation tool can then provide the required inputs to achieve a user-specified desired performance. This open-source tool allows researchers to train design automation tools on different microfluidic components such as droplet generators and droplet sorters.

Performance prediction

Machine learning techniques enable system behavior prediction and have advanced several fields from biology to cyber-physical systems [9]. With the introduction of low-cost microfluidic fabrication in recent years, large design spaces previously only explored using numerical simulations [12], can now be studied experimentally in a time- and cost-efficient manner [5]. Therefore, machine learning algorithms can now be trained on large and reliable experimental data-sets to generate accurate predictive models for different microfluidic modules. Several built-in machine learning algorithms are included in DAFD and the user can pick the most accurate model for the experimental data of any given microfluidic, as shown in Fig. 2.

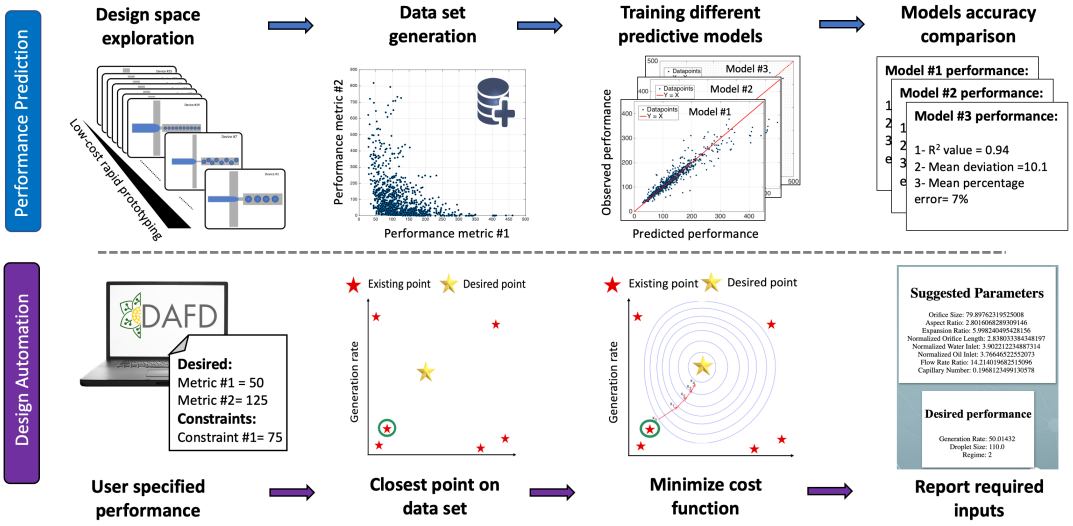


Figure 2: Large experimental datasets can now be generated using low-cost micromilling. Once a dataset is generated, several built-in machine learning algorithms (neural networks, random forest, support vector regression, nearest data-point, etc.) can be trained on the dataset to develop an accurate predictive model. The reverse model exploits the predictive model, dataset, and gradient descent to provide the required inputs for the desired performance. Thus, eliminating the need for design iterations.

Design automation

Once an accurate predictive model is generated, DAFD workflow allows the user to specify the desired performance and constraints. As shown in Fig. 2, the algorithm starts with finding a point on the data-set that is closest to the specified performance. Then, by defining a cost function Eq. (1) and using gradient descent the cost function is minimized until the desired performance is reached and the inputs that resulted in that performance are reported.

$$F(x) = \sum_{i=1}^{n} (P_{i,d} - P_{i,x})^2 \quad (1)$$

where $F(x)$ is the cost function, $P_{i,d}$ is the i_{th} desired performance metric, and $P_{i,X}$ is the i_{th} current performance.

3 CONCLUSION AND FUTURE WORK

DAFD is a modular design automation tool for microfluidics that enables researchers to design microfluidic devices by specifying the desired performance. Thus, lowering the barrier to entry to microfluidics for life science groups with limited knowledge of microfluidics. DAFD allows for experimental researchers with limited knowledge of machine learning to build design automation tools by training DAFD forward models on an input-performance data-set of any microfluidic component through an open-source codebase.

REFERENCES

[1] Lashkaripour, A., Abouei Mehrizi, A., Rasouli, M., and Goharimanesh, M. Numerical study of droplet generation process in a microfluidic flow focusing. *Journal of Computational Applied Mechanics 46*, 2 (2015), 167–175.

[2] Lashkaripour, A., Goharimanesh, M., Mehrizi, A. A., and Densmore, D. An adaptive neural-fuzzy approach for microfluidic droplet size prediction. *Microelectronics Journal 78* (2018), 73–80.

[3] Lashkaripour, A., Mehrizi, A. A., Goharimanesh, M., Rasouli, M., and Bazaz, S. R. Size-controlled droplet generation in a microfluidic device for rare dna amplification by optimizing its effective parameters. *Journal of Mechanics in Medicine and Biology 18*, 01 (2018), 1850002.

[4] Lashkaripour, A., Rodriguez, C., and Douglas, D. A reverse predictive model towards design automation of microfluidic droplet generators. International Workshop on Bio-Design Automation (IWBDA).

[5] Lashkaripour, A., Rodriguez, C., Ortiz, L., and Densmore, D. Performance tuning of microfluidic flow-focusing droplet generators. *Lab on a Chip 19* (2019), 1041–1053.

[6] Lashkaripour, A., Sanka, R., Lippai, J., and Densmore, D. Design automation based on fluid dynamics. International Workshop on Bio-Design Automation (IWBDA).

[7] Lashkaripour, A., Silva, R., and Densmore, D. Desktop micromilled microfluidics. *Microfluidics and Nanofluidics 22*, 3 (2018), 31.

[8] Lippai, J., Sanka, R., Lashkaripour, A., and Densmore, D. Function-driven, graphical design tool for microfluidic chips: 3duf. International Workshop on Bio-Design Automation (IWBDA).

[9] Mehdipour, N., Briers, D., Haghighi, I., Glen, C. M., Kemp, M. L., and Belta, C. Spatial-temporal pattern synthesis in a network of locally interacting cells. In *2018 IEEE Conference on Decision and Control (CDC)* (2018), IEEE, pp. 3516–3521.

[10] Ortiz, L., Lashkaripour, A., and Densmore, D. Automating functional enzyme screening & characterization. International Workshop on Bio-Design Automation (IWBDA).

[11] Rasouli, M., Abouei Mehrizi, A., and Lashkaripour, A. Numerical study on low reynolds mixing oft-shaped micro-mixers with obstacles. *Transp Phenom Nano Micro Scales 3*, 2 (2015), 68–76.

[12] Rasouli, M., Mehrizi, A. A., Goharimanesh, M., Lashkaripour, A., and Bazaz, S. R. Multi-criteria optimization of curved and baffle-embedded micromixers for bio-applications. *Chemical Engineering and Processing-Process Intensification 132* (2018), 175–186.

Accelerating the Threshold and Timing Analysis of Genetic Logic Circuit Models

Sanaullah*
Chosun University
Gwangju, South Korea
mr.raosanaullah@gmail.com

Hasan Baig
Habib University
Karachi, Pakistan
hasan.baig@sse.habib.edu.pk

Jeong A Lee
Chosun University
Gwangju, South Korea
jalee@chosun.ac.kr

ABSTRACT

Synthetic genetic logic circuits are an application of synthetic biology where biological parts of the DNA from inside a cell are engineered to implement logical functions. Such circuits exhibit a nondeterministic behavior and are much harder to characterize in contrast to electronic logic circuits. Approaches to analyze the threshold values and timing analysis of single or cascaded genetic logic circuits have been previously proposed, however, they suffer from long latency. There is a potential for parallelizing the procedure which verifies the obtained threshold values for obtaining lower run times. In this work, we introduce a fast approach to estimating threshold values through a parallelized verification procedure. The proposed algorithm takes significantly less time than the previous approach exhibiting a time complexity of $\emptyset(N)$ reduced from $\emptyset(N^2)$.

1 INTRODUCTION

Dynamic Virtual Analyzer and Simulator or D-VASim [1] is an interactive virtual laboratory environment for the simulation and analysis of genetic logic circuit models represented in System Biology Markup Language or SBML [2]. D-VASim helps replace the time-consuming in-vitro experiments (laboratory experiments) needed to analyze and design genetic circuits. D-VASim can take up to hours in estimating the threshold values for complex genetic circuits [3] owing to its serial execution necessitated by data dependency. Such simulation times are still acceptable compared to the number of days required to spend in laboratory experiments for only a single input combination and specific parameter set. However, there is potential for much faster approaches to genetic circuit simulation. D-VASim applies both deterministic and stochastic analyses to extract and validate Boolean logic from the SBML model.

The input to D-VASim is genetic circuit model where D-VASim allows the user to interact with the model, observe its behavior, and make direct changes in the concentration of input protein(s) at run-time. D-VASim is capable of estimating the threshold value for a genetic circuit model [1]. The threshold value refers to the minimum input protein(s) concentration that causes the average output protein(s) concentration to cross the input protein(s) concentration. The Boolean function of a genetic logic circuit can be verified by extracting the logic behavior from the simulation data. Thus threshold value and timing analysis are used to verify a genetic logic circuit's intended boolean function.

This methodology is useful in helping the user extract the Boolean logic of the bio-models without any prior knowledge of the expected behavior of the model. The proposed approach is similar to the previous approach [3] in a way that it also estimates the single threshold value required by all inputs to trigger the final output. However, as compared to the previous version of D-VASim which takes too long to produce the estimations for complex circuits owing to its sequential execution and data dependencies, the proposed approach eliminates the data dependency and applies multi-threading and loop unrolling in the threshold value verification procedure leading to a faster threshold value estimation.

2 PROPOSED ALGORITHM

The improved algorithm for threshold value analysis of genetic logic circuit is shown in Algorithm 1. The parameter definitions are also presented as comments in Algorithm 1. More details on the function that each parameter represents can be obtained from [3]. The algorithm is initialized by user-defined parameters. We have used the same parameter values as used in [3]. Lines $11 - 19$ verify the obtained threshold value by iterating the model for a predefined number of iterations I. In the previous algorithm [3], due to a sequential verification process implementation, the average of the running array is computed post-verification. In comparison, we compute the average of every single array for only those elements which are above the possible threshold value (PT). This average is used for Estimate Consistency E and then checked against the user-defined parameters for the upper and lower threshold values. The way we implement the verification process eliminates any data-dependency and all the required iterations can be run in parallel significantly reducing latency.

Another reason the proposed verification process is faster compared to [3] is because we filter the input concentration values greater than PT and we only use these values to compute the running average in each iteration. In Figure 1, A shows the task flow of the algorithm from [3] and B shows the task flow of the proposed algorithm. We can clearly see that the proposed algorithm runs all the iterations i in the verification process simultaneously. In each iteration, the running average is calculated after filtering the input concentration values. Thus the time complexity can be calculated for [3] as $N_{outer}(T_{p1} + (N_{inner}T_{p2}) + T_{p3})$ and $N_{outer}(T_{p1} + T_{p2} + T_{p3})$ for the proposed algorithm. In the case where user inputs $N_{outer} = N_{inner} = N$, then, the time complexity for [3] becomes $\emptyset(N^2)$ and for the proposed algorithm $\emptyset(N)$.

Algorithm 1: Threshold Value Analysis

```
1  begin
2  INITIALIZE (I_c, F_c, O_s, O_t, OC_DUTH, OC_DLTH, OC_DLTH, Inc,
     S_t, N)
     /*
     I_c = Initial input concentration where the analysis should start from
     F_c = Final input concentration where the analysis should stop at
     Inc = Increment Value: The value which is added to the previous
     concentration level until the concentration level reaches to
     O_s = Name of output specie
     O_t = Initial time to check selected circuit behavior
     S_t = Settling time
     OC_DUTH and OC_DLTH = User defined percentage acceptance of
     output consistency for upper and lower threshold value respectively.
     N = Number of iterations to verify the consistency of results
     V_t = Time to verify each iteration N
     */
3    for all possible combination do
4      if (I_c == 0 ) then
5        Determine initial Output concentration
6    while (I_c <= F_c) do
7      if (O_c > I_c) then  /* O_c Output of concentration of selected specie */
8        PT = I_c  /* PT = Possible Threshold value */
9        /* Verification Process */
10       for i from 1 to N parallel do
11         while (T1 <= V_t) do
12           Execute Simulation
13           if (T_c2 > S_T)
14             /* Filter Output */
15             if input concentration > PT then
16               Store this output concentration in array (j)
17           end
18           Take the running average of array (j)
19       end
20       Estimate consistency E_c
21       if (E_c >= OC_DUTH) then
22         Consider lower threshold value = 0, if not found already
23         Return the results and terminate all loops
24       else
25         if (E_c <= OC_DLTH) then
26           Save lower threshold value and resume analysis
27         else
28           Resume analysis
29       I_c = I_c + Inc
30       T1 = 0
31   end
32 end
```

Table 1: Results Comparison

Genetic Logic Circuit	Threshold Value Analysis	
	[1] (sec)	***Proposed Algorithm (sec)***
Ckt1 NOT	17	0.17
Ckt2 NAND	32	0.37
Ckt3 AND	80	0.62
Ckt4 NOR	15	0.12
Ckt5 OR	40	0.72
Ckt6 Delay	60	0.51

Results generated using D-VASim

3 RESULTS

The performance of the proposed algorithm is compared with the one implemented in current version freely available online [4]. We run experiments on the set of genetic logic circuit models presented in [5]. The threshold analysis run-times from D-VASim and our algorithm are listed in Table 1. It is clear from Table 1 that with the proposed modification to the verification step, the threshold value estimation times are significantly reduced.

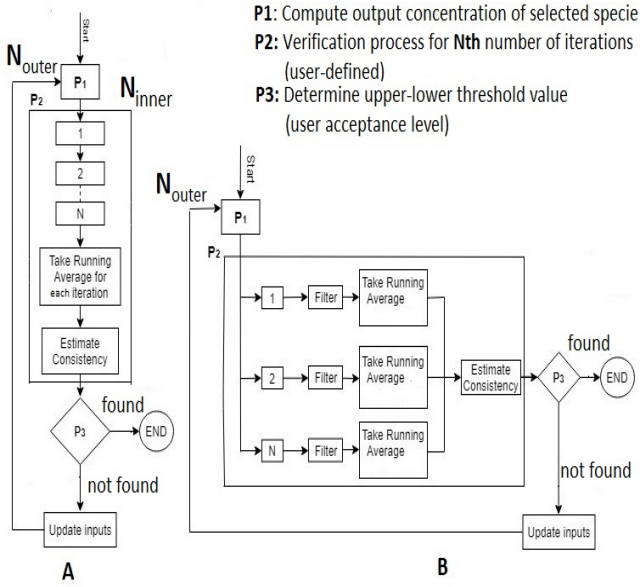


Figure 1: Task flow of (A) Previous algorithm [3] and (B) The proposed algorithm.

CONCLUSION

In this work, we have proposed modification to accelerate a state-of-the-art threshold analysis procedure and subsequently the timing analysis of genetic logic circuits. The main contribution lies in paralellizing the threshold value verification process. We achieved significant reduction in threshold value analysis latency by a huge factor in all of the test cases.

ACKNOWLEDGEMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2019R1I1A3A01058887) and in part by the Korea Institute of Energy Technology Evaluation and Planing and Ministry of Trade, Industry and Energy of the Republic of Korea under Grand 20184010201650..

REFERENCES

[1] Hasan Baig and Jan Madsen. D-VASim: An Interactive Virtual Laboratory Environment for the Simulation and Analysis of Genetic Circuits. *Bioinformatics*, 33(2):297–299, 2017.

[2] Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[3] Hasan Baig and Jan Madsen. Simulation Approach for Timing Analysis of Genetic Logic Circuits. *ACS synthetic biology*, 6(7):1169–1179, 2017.

[4] D-vasim (v1.2). http://bda.compute.dtu.dk/downloads/d-vasim/. v1.2, October 2016.

[5] Chris J Myers. *Engineering Genetic Circuits.* Chapman and Hall/CRC, 2016.

Better research by efficient sharing: evaluation of free management platforms for synthetic biology designs

Uriel Urquiza-García[1*]
Tomasz Zieliński[2*]
jurquiza@ed.ac.uk
tomasz.zielinski@ed.ac.uk
[1]Institute for Molecular Plant Sciences, D. Rutherford Building, University of Edinburgh, King's Buildings, EH9 3BF
Edinburgh, U.K.

Andrew J. Millar[2]
[2]SynthSys and School of Biological Sciences, C. H. Waddington Building, University of Edinburgh, King's Buildings, EH9 3BF
Edinburgh, U.K.
Andrew.Millar@ed.ac.uk

1 INTRODUCTION

A top goal in synthetic biology is incorporating engineering principles into biology, for example through modular approaches in gene assembly. This approach requires adequate resources for managing data in order to speed up the design-build-test-learn cycle. We performed an evaluation of two free open source software platforms for data management in synthetic biology: JBEI-ICE [1] and Synbiohub [2]. The analysis was done from the standpoint of experimental biology research groups in academia, which are interested in using these types of repositories for managing their synthetic biology data. We defined a minimal set of requirements for repositories in this context and explored the platforms performance in three real world scenarios: 1) support for the design-build-test-learn cycle, 2) batch submission of existing design into repositories and 3) Re-use and discovery of designs present in repositories. This work provides an insight into the current state of synthetic biology resources, we hope that it will encourage wider tool adoption and help guide future development that will impact on design automation.

2 METHODOLOGY

The public instances of JBEI-ICE and Synbiohub were evaluated by performing tasks that are executed in experimental labs. I order to explore the use cases, the team worked together with experimentalist in order to document different data-management scenarios. The evaluation was distilled into tabular form of features. Therefore we provide a useful guideline for interested new users or developers benchmarking their applications.

*Both authors contributed equally to this research. This work was funded by the Wellcome Institutional Strategic Support Fund (ISSF3) and the UK Centre for Mammalian Synthetic Biology (award BB/M018040/1)

3 RESULTS

Tables generated in this work include:

- Minimal requirements for a parts registry.
- Requirements for collaborative design-build-test-learn research cycle.
- Requirements for batch operations
- Examples of search use-cases

We present Table 1 one as example, from our just accepted manuscript [3].

Table 1: Minimal repository requirements

Features	Synbiohub	ICE
Unique identifiers	yes	yes
Descriptive name	yes	yes
Storing exact sequence	yes	yes
Access permissions	yes	better
Visualize sequence map	yes	yes
Import common sequence formats	yes	yes
Attributions	yes	better
User labels and tags	yes	yes
Functional categorization (e.g. promoter, tag)	better	yes
Device level categorization (e.g. logic gate)	yes	yes
Chassis	no	yes
Bibliography information	better	yes
Licence information for parts	no	yes

REFERENCES

[1] Ham, T. S., Dmytriv, Z., Plahar, H., Chen, J., Hillson, N. J., and Keasling, J. D. Design, implementation and practice of jbei-ice: an open source biological part registry platform and tools. *Nucleic Acids Res 40,* 18 (Oct 2012), e141.

[2] McLaughlin, J. A., Myers, C. J., Zundel, Z., Misirli, G., Zhang, M., Ofiteru, I. D., Goñi-Moreno, A., and Wipat, A. Synbiohub: A standards-enabled design repository for synthetic biology. *ACS Synth Biol 7,* 2 (02 2018), 682–688.

[3] Urquiza-García, U., Zieliński, T., and Millar, A. Better research by efficient sharing: evaluation of free management platforms for synthetic biology designs. *Oxford Synthetic Biology* (June 2019).

A method for compiling arbitrary transfer functions to molecular circuits

Iuliia Zarubiieva
University of Warwick
Coventry CV4 7AL, UK
i.zarubiieva@warwick.ac.uk

Francesca Mantellino
University of Warwick
Coventry CV4 7AL, UK
francesca.mantellino@warwick.ac.uk

Andrew Phillips
Microsoft Research
Cambridge CB1 2FB, UK
andrew.phillips@microsoft.com

Vishwesh Kulkarni
University of Warwick
Coventry CV4 7AL, UK
v.kulkarni@warwick.ac.uk

1 INTRODUCTION

As an engineering substrate, DNA is well suited for the construction of biochemical circuits and systems due to its simple design based on the Watson-Crick paring rules [1]. *DNA Strand Displacement* (DSD) is an implementation strategy based on the hybridization of DNA strands with partial or full complementarity, resulting in the displacement of one or more pre-hybridized strands [1], [2], [3]. `Visual DSD` is a *graphical user interface* (GUI) aided software platform that enables an *in silico* design and simulation of DSD-based circuits [4], [5]. The platform accounts for system complexity and potential unwanted interferences between molecules in the system and is intended to minimine the gap between the theoretical simulation results obtained *in silico* and the experimental results obtained in the wet lab. If a given system is modelled as either a set of DSD reactions or as an *Abstract Chemical Reaction Network* (ACRN) then it can be simulated by running the corresponding text file in, respectively, the "DSD" or the "CRN" mode of `Visual DSD`, which allow interactions of domains with one or more mismatched bases [4]. The end results of the DSD systems is quite sensitive to the values of parameters – typically, these include the reaction rates, concentrations, the degree of complementarity, etc.

Hardware description languages (HDLs) enables a user to design an electronic system through textual commands that are transformed into a physical implementation of the circuit in silicon. Recently, `Cello` has applied this approach to genetic circuits to transform an HDL design into a linear DNA sequence that can be constructed and run in living cells [6]. However, `Cello` does not facilitate the design of computational nucleic acid devices. In [7], a general purpose CRN-to-DSD compiler `Nuskell` has been developed and its benefit has been illustrated through interesting applications [7]. Unlike `Nuskell` which uses a top-down approach, our complier uses a bottom-up approach that makes use of domains as starting building blocks, then connects them into strands, describes the interaction rules for different types of reactions, combines reactions into blocks which together generate the full circuits.

The current work focuses on the optimisation of `Visual DSD` through automation of the code generation and simplification of the introduction of `Visual DSD` to a wider reaserch comunity. The designed compiler will relax the requirement that the user should be proficient in `Visual DSD` programming language.

2 OUR COMPILERS: RATIONALE AND OVERVIEW

Any *linear time-invariant* (LTI) system S can be represented by a so-called *transfer function* (TF), which is a frequency-domain representation. This, in turn, can be expressed as a connection of a finite number of integrators, scalar gains, and summation blocks, as shown in Figure 1(C and D). Each of these component blocks can be realised by finitely many CRN's which comprise catalysis, annihilation, and degradation reactions. Each of these component blocks can also be realised through a well-known set of DSD reactions. In order to optimise the reaction parameters, we have designed an algorithm based on *particle swarming optimisation* (PSO), summarised in [8]. This can be used additionally to update the default parameters set by the compiler. So, we have synthesized two MATLAB-based compilers (1) `TF-to-CRN`: its output is a text file of a ACRN representation of S which should be run in the "CRN" mode of `Visual DSD` and (2) `TF-to-DSD`: its output is a text file of a DSD representation of S which should be run in the "DSD" mode of `Visual DSD`. Our compilers support 2-domain, 3-domain, and 4-domain representations: in general, the representation is called k-domain if each chemical species is implemented as a single strand of DNA comprising k distinct domains. The operation of our `TF-to-DSD` compiler can be summarised as follows; `TF-to-CRN` can be explained similarly:

(1) **Step 1:** In the GUI of `TF-to-DSD`, the user inputs TF of the LTI system that they want to design.

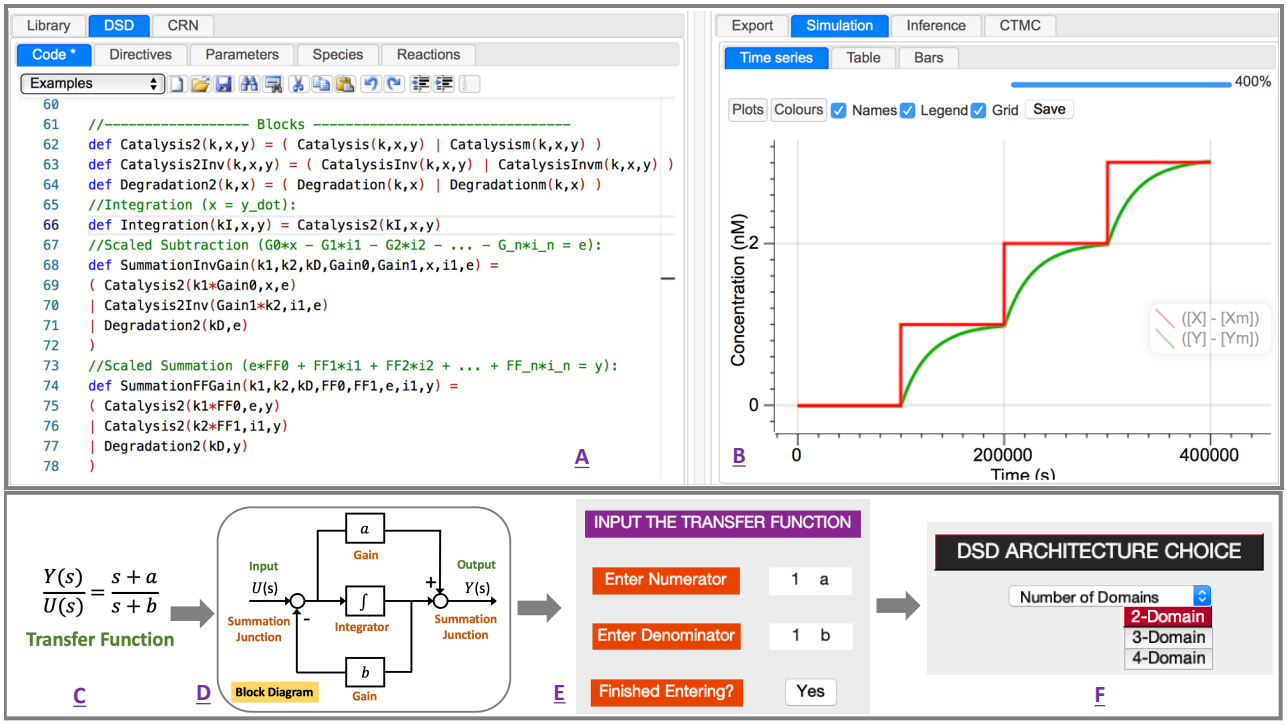


Figure 1: Our `TF-to-DSD` compiler in action. A: This `Visual DSD` screenshot illustrates the DSD code to implement a *low-pass filter* (LPF), given in C, generated by our `TF-to-DSD` compiler. B: The output of this LPF when excited with a staircase input – the red color plot represents the input and the green color plot represents the output. D: In the generated code, all necessary blocks such as "Scaled Summation" (see line # 73) can be created and modified on the fly. After entering the polynomial coefficients as illustrated in E, the user has a choise of the type of DSD architecture F.

(2) **Step 2:** In the GUI of `TF-to-DSD`, the user inputs the number of domains of the desired DSD architecture.
(3) **Step 3:** `TF-to-DSD` computes a minimal state-space model M for this TF; M comprises a finite number of integrators, scalar gains, and summation blocks.
(4) **Step 4:** For each block, building on [2], `TF-to-DSD` creates a text file containing (1) the DNA strand compositions and (2) a minimal set of DSD reactions.
(5) **Step 5:** Using PSO, `TF-to-DSD` optimises all DSD parameters such as the concentrations, chemical reaction rates, the degree of complementarity.

The resulting text file can now be run in the "DSD" tab of `Visual DSD` for the simulation and analysis purposes, and to provide the initial data for the wet-lab implementations.

3 RESULTS AND DISCUSSION

As illustrated in Figure 1, our compiler gives satisfactory results. Also, we have shown how the bottom-up architecture of `Visual DSD` can be adequately generalised to implement not only different algorithms but also to synthesize new conceptual modules on the fly for LTI systems. Our approach can be generalised for dynamic nonlinear systems as well if the input is presented in a different way, for example, by a block diagram. Besides speeding up the *in silico* design of DSD-based circuits, our compilers also increase the outreach of DNA computation to users who might only be comfortable with mathematical models and MATLAB/Python.

Acknowledgments

This research is supported, in parts, by the EPSRC INDUSTRIAL CASE AWARD (CASE Voucher 16000070), Microsoft Research, the EPSRC/BBSRC grant BB/M017982/1 to the Warwick Integrative Synthetic Biology Centre, and the Erasmus+ grant to the University of Warwick (Agreement # 2018-1-UK01-KA107-047454).

REFERENCES

[1] F. Simmel *et al*, Chem. Rev., Feb 2019, Article ASAP.
[2] D. Soloveichik *et al*, PNAS, March 2010; 107(12): 5393-5398.
[3] T. Song *et al*, ACS Syn. Bio., 2016; 5(8):898-912.
[4] B. Yordanov *et al*, ACS Syn. Bio., 2014; 3(8):600-616.
[5] C. Spaccasassi *et al*, Visual DSD User Manual, Microsoft Research, Cambridge, UK, 2019.
[6] A. Nielsen *et al*, Science, Apr 2016; 352(281): aac7341.
[7] S. Badelt *et al*, Int. Conf. DNA-Based Computers, 2017.
[8] Y. Zhang *et al*, Math. Problems in Eng, 2015; 2015, Article ID 931256.

The Morphogen Circuit Builder & Compiler

Bryan Bartley[1], **Brian Basnight**[1], **Jesse Tordoff**[2], **Jacob Beal**[1], **Ron Weiss**[2]
[1]Raytheon BBN Technologies, [2]Massachusetts Institute of Technology
bryan.a.bartley@raytheon.com

1 INTRODUCTION

Engineering cells to achieve programmed development of complex three-dimensional shapes (also known as morphogenesis) would provide a powerful means of tissue engineering for application areas such as regenerative medicine and drug development. Morphogenesis is a complex process involving cell-to-cell signaling, biomechanical forces, cell division, and differential adhesion. In order to program cells to organize themselves into the diverse types of structures seen in nature, as well as entirely novel structures, synthetic biologists need a toolkit of morphogenetic parts and means of composing those parts to produce predictable geometric forms. We have thus been investigating genetic "building blocks" for engineered morphogenesis and have developed the "Morphogen Circuit Builder," a computational tool that allows us to explore a large space of designs that can be implemented with these parts, and a "Morphogen Compiler" that uses the circuit builder to generate a such designs from a high-level description of target morphologies.

2 MORPHOGEN CIRCUIT BUILDER & COMPILER

The Morphogen Circuit Builder is a high-level Java API that represents circuit designs using the Synthetic Biology Open Language format (SBOL). Leveraging SBOL allows us to compose hierarchical designs from abstract genetic modules by connecting regulatory inputs, outputs, and recombination targets. Abstract modules are then instantiated by selecting parts from an inventory of DNA parts, including promoters, regulatory elements, recombinases, recombination sites, a variety of cadherin coding sequences, reporters, and degradation tags. The circuit builder stores and accesses these virtual parts in a SynBioHub repository [3] populated by scraping GenBank sequences from vector maps and converting them to SBOL format. A key step in the process was annotating these modules with interaction information and descriptive ontology terms so that the compiler can automatically select and appropriately match modules together.

The circuit builder can also transform multi-layered, modular designs into one or more target sequences. This involves flattening the modular, hierarchical design to a single layer and unifying all input/output connections. Once this is accomplished, all of the sequences of sub-components are concatenated together into a single sequence for each plasmid to be assembled and delivered.

The circuit builder also automatically generates a two-dimensional model of morphogenetic differentiation and sorting. These models are based on the Cellular Potts Model

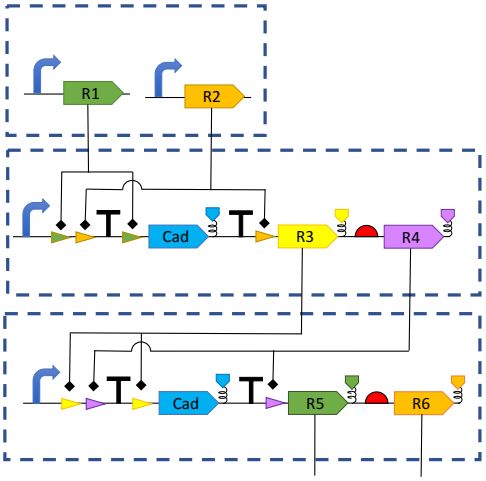


Figure 1: Switch modules can be chained in order to program multi-stage differentiation.

and can be simulated using CompuCell3D [5]. Although a variety of cell modeling frameworks could be applicable, CompuCell3D's programmatic API enables integration with Morphogen Compiler and includes plugins for simulating different processes related to morphogenesis (e.g., diffusion gradients, division). Characteristic parameters, such as expression level or recombination efficiency, are associated with modules in the repository, which then determine the corresponding parameters in the model. During compilation, as one module is connected to another, the compiler performs operations that augment the model depending on which type of modules are being connected. Furthermore, by exploring a variety of circuit designs, we have established a predicted phase space for certain morphogenetic systems, which has then allowed us to implement the Morphogen Compiler. While the examples here are based specifically on a readily available library of parts controlling differential adhesion, our toolchain can be extended to apply to other morphogenetic processes as parts become available.

The Morphogen Compiler generates a circuit design based on a user-specified, high-level description of a morphology. The compiler is capable of generating designs for multiple types of morphologies, such as multi-layer, concentric sorted balls (analogous to gastrulation) or specialized, cell clusters interspersed through tissue (analogous to clusters of endocrine cells in pancreatic tissue) [4]. The user specifies the desired morphology and target characteristics such as diameter, layer thickness, average cluster size, and/or number of clusters. Morphologies are then generated by controlling the probability with which cells differentiate into different

types, as well expression levels of one or more types of cadherins (a family of cell-adhesion membrane proteins playing a key role in tissue morphogenesis, leading to cell sorting through differential adhesion [2]). The compiler then generates a circuit design by selecting modules with appropriate parameters, then proceeds to realization and/or simulation via the circuit builder as described above.

3 RESULTS

We have tested the compiler and circuit builder with a variety of differentiation cascade circuits. These cascades are composed of a modular recombinase-based switch that differentiates a region probabilistically into either of two cell fates, one of which expresses an adhesion protein while the other produces additional recombinase outputs that can then be chained to another downstream differentiation switch. This motif based design thus allows specification of structures with an arbitrary number of cell types. Figure 1 shows how a differentiation switch is implemented and how multiple switches can be chained in a cascade.

Differentiation into one cell fate versus the other is governed by competition between a pair of recombinases for their cognate recombination sites. The probability of one recombination event versus the other depends on the relative expression levels of each recombinase, which is modulated through the use of degradation tags, which have previously been demonstrated for fine-tuning circuits [1].

The differentiation switches are parameterized with a differentiation probability (f) that determines the relative probability of selecting a lower-adhesion cell fate vs. a higher-adhesion cell fate: at each stage, low relative f will tend to result in complete cell sorting, while high f will instead tend to result in quasi-stable cluster formation. Cadherin modules expression levels are then modulated to achieve the parameterized degree of adhesion means by degradation tags, thus allowing an exponential number of different morphogenetic patterns to be generated from differential tuning of a single composable motif. Figure 2 shows examples (using CompuCell3D simulation of compiler-generated models) of different morphologies that can be generated by modulation of the number of stages and the level of f for each stage.

4 FUTURE WORK

Our future goals are to explore a wider range of possible morphogenetic shapes by optimizing simulation parameters to better produce desired topological characteristics and match these more closely to laboratory results. To support this goal, we have implemented algorithms in the simulator that calculate image-based, topological parameters, such as cluster number, cluster size, and heterotypic boundary length (a measure of cell sorting). Additional metrics, such as image symmetry will need to be considered as well. In addition,

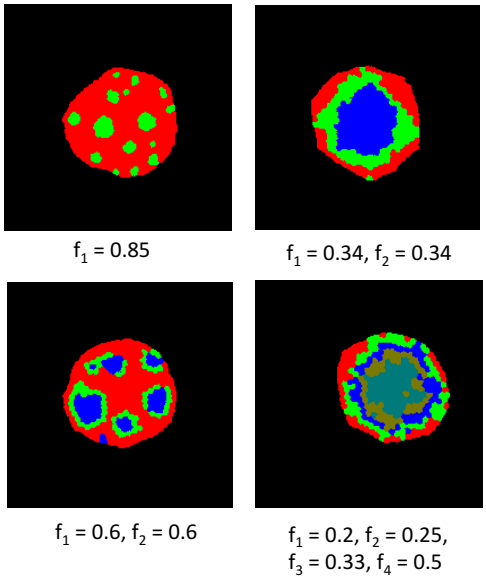


Figure 2: Controlling probability of switching into cell types with different cadherin expression profiles can generate different morphologies, as shown in these CompuCell3D[5] simulations generated by the Morphogen Compiler.

we have parallelized simulations so that we can efficiently explore large parameter spaces. These features will allow us to identify optimal parameter sets that generate a variety of target morphologies, including asymmetric morphologies.

5 ACKNOWLEDGEMENTS

This work has been supported by the Defense Advanced Research Projects Agency under Contract No. W911NF-17-2-0098. The views, opinions, and/or findings expressed are of the author(s) and should not be interpreted as representing official views or policies of the Department of Defense or the U.S. Government. This document does not contain technology or technical data controlled under either U.S. International Traffic in Arms Regulation or U.S. Export Administration Regulations.

REFERENCES

[1] Chassin, H., Müller, M., Tigges, M., Scheller, L., Lang, M., and Fussenegger, M. A modular degron library for synthetic circuits in mammalian cells. *Nature communications 10*, 1 (2019), 2013.

[2] Foty, R. A., and Steinberg, M. S. The differential adhesion hypothesis: a direct evaluation. *Developmental biology 278*, 1 (2005), 255–263.

[3] McLaughlin, J. A., Myers, C. J., Zundel, Z., Misirli, G., Zhang, M., Ofiteru, I. D., Goni-Moreno, A., and Wipat, A. SynBioHub: a standards-enabled design repository for synthetic biology. *ACS synthetic biology 7*, 2 (2018), 682–688.

[4] Minami, K., Okano, H., Okumachi, A., and Seino, S. Role of cadherin-mediated cell-cell adhesion in pancreatic exocrine-to-endocrine transdifferentiation. *Journal of Biological Chemistry 283*, 20 (2008), 13753–13761.

[5] Swat, M. H., Thomas, G. L., Belmonte, J. M., Shirinifard, A., Hmeljak, D., and Glazier, J. A. Multi-scale modeling of tissues using compucell3d. In *Methods in cell biology*, vol. 110. Elsevier, 2012, pp. 325–366.

Machine Learning Algorithms for Robust Meta-Analysis of Gene Expressions

Vishwesh Kulkarni
University of Warwick
Coventry CV4 7AL, UK
v.kulkarni@warwick.ac.uk

Xinwu Yu
University of Warwick
Coventry CV4 7AL, UK
X.Yu.8@warwick.ac.uk

Weikang Qian
Shanghai Jiao Tong University
Shanghai 200240, China
qianwk@sjtu.edu.cn

INTRODUCTION

A fuller benefit from the enormous amounts of data generated by omics technologies, as evident from the 1000 Genomes Project, Wellcome Trust Case Control Consortium, Exome Aggregation Consortium and others, can be realised only if a repository of common reference datasets together with efficient meta-analysis techniques is developed [1], [2]. In genomics alone, the amount of publicly available gene expression data is enormous: (1) NCBI GEO has over 3 million samples collected from more than 19,000 platforms and (2) ArrayExpress has 54.46 TB worth data from more than 72,000 experiments conducted on more than 2.4 million assays.

Use of *elastic net* [3] in meta-analysis was proposed by Hughey-Butte in [2] for classification and regression purposes, and its efficacy was demonstrated via cancer datasets. Using 629 samples from five data sets, they trained a multinomial classifier to distinguish between four lung cancer subtypes. Their meta-analysis-derived classifier included 58 genes and achieved an excellent accuracy of 91% on leave-one-study-out cross-validation and on three independent data sets. In this manuscript, we extend the approach of [2] to show how meta-analysis using gradient-boosted decision trees can realize an even greater classification accuracy.

MATERIALS AND METHODS

We used the discovery dataset D_0 created in [2]. This dataset contains curated data from eight publicly available microarray studies of lung cancer. Four cancer subtypes are of interest: *adenocarcinoma* (AD), *squamous cell carcinoma* (SQ), *small cell lung carcinoma* (SCLC), or *carcinoid* (CAR). In this dataset, the number of samples is 639 whereas the number of genes is 7,200. Thus, this is a high-dimensional dataset, with the number of predictors being an order of magnitude larger than the number of samples. Our classifier model is trained on the merged discovery dataset using machine learning approaches and then tested separately on each of the validation dataset as well as on the merged validation dataset.

For training the model, given the high-dimensional nature of the dataset, elementary machine learning methods cannot be used directly owing to the "curse of dimensionality" [3], [4]. Ensemble methods such as *Random Forest* (RF) and *Gradient Boosted Trees* (GBT) are helpful in solving such problems since these create aggregate of individual models: the individual models must be designed such that those models do not suffer from the curse of dimensionality [5], [6], [7]. The aggregation is accomplished via *bagging* and *boosting* techniques – bagging refers to aggregating the individual models in parallel whereas boosting refers to aggregating those sequentially. RF is an example of the methods that use bagging while GBT is an example of the methods that use boosting. For validation, three datasets are used – each corresponding to one of the studies, viz., CLCGP, GSE37745, and GSE41271. We trained the classifer model on the merged discovery dataset and then tested it separately on each of the validation dataset as well as a merged validation dataset.

Our synthesis of the classifier, and its verification, can be summarised as follows for the GBT approach; the case for RF can be summarised on similar lines:

(1) **Step 1 – Create a standardised training dataset** D: The corresponding cancer type was found out from the sample metadata and the column of cancer types was combined with the gene expression data to form the training dataset D for the model.
(2) **Step 2 – Train the classifier on** D **using GBT:** Use the R package `xgboost` (see [8]) with the following configuration: (i) # iterations = 9, (ii) cross-validation folds = 5, (iii) evaluation matrix: log-loss, and (iv) objective function: multiclass softmax.
(3) **Step 3 – Fit the model:** Fit and test the model independently for each validation dataset. Furthermore, to check the overall accuracy of the model, merge the three validation datasets and then validate the model on the merged validation datasets.

To ensure reproducibility, a random seed is set in Step 3 prior to each fitting.

RESULTS AND DISCUSSION

Replacing the use of elastic net with RF resulted in no improvement in the results reported in [2]. However, replacing the use of elastic net with GBT led to fairly significant improvements as follows. The confusion matrices obtained in the resulting 5-fold cross-validations are shown in Figure 1; in a confusion matrix, the columns represent the predicted

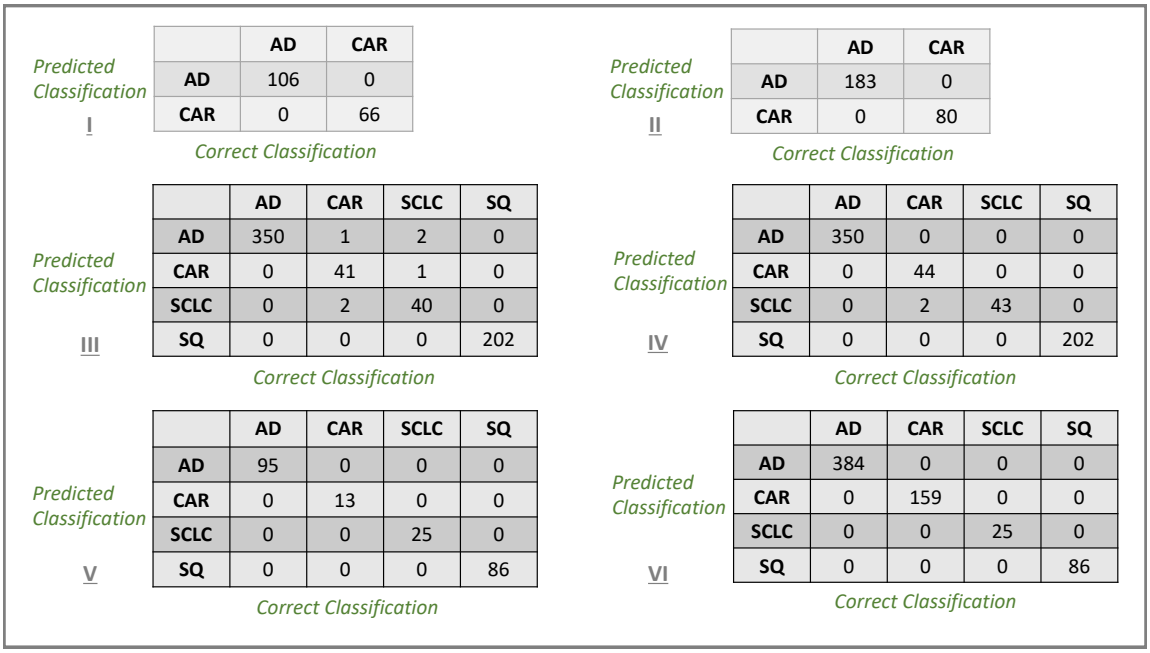


Figure 1: Confusion matrices generated by our GBT-based meta-analysis method on the datasets studied in [2].

values and the rows represent the actual values. Figure 1(I) gives the confusion matrix for the 5-fold cross-validation on the dataset corresponding to GSE37745. This dataset has 172 samples: 106 of type AD and 66 of type CAR. Here, our predictive accuracy is 100%. Figure 1(II) gives the confusion matrix for the 5-fold cross-validation on the dataset corresponding to GSE41271. Here, our predictive accuracy is 100%. Figure 1(III) shows the confusion matrix for the 5-fold cross-validation on the merged discovery dataset: predictive accuracy is 99.06% which is more than the 91.2% accuracy reported in [2]. Figure 1(IV) shows the confusion matrix for the 5-fold cross-validation on the entire merged discovery dataset: predictive accuracy is 100% – here, the perfect accuracy might be an indication of over-fitting. Figure 1(V) gives the confusion matrix for the 5-fold cross-validation on the dataset corresponding to CLCGP. Here, our predictive accuracy is 88.58% which is slightly better than the figure of 86% achieved in [2]. Figure 1(VI) gives the confusion matrix for the 5-fold cross-validation on the merged validation dataset comrpising 654 samples. Here, our predictive accuracy is 96.18% which is better than the 91.3% accuracy achieved in [2]. This accuracy is realised in just 9 iterations, resulting in savings in computational cost as well. The superior performance on all individual validation datasets as well as on the merged validation dataset reveals that there is no overfitting. Furthermore, we have identified X9997, X11127, X11166, X3920, and X22920 as the five most important predictors with X9997 being overwhelmingly important.

ACKNOWLEDGMENTS

This research is supported, in parts, by the EPSRC/BBSRC grant BB/M017982/1 to the Warwick Integrative Synthetic Biology Centre and the Erasmus+ grant to the University of Warwick (Agreement # 2018-1-UK01-KA107-047454).

REFERENCES

[1] K. Zalocusky *et al. Cell Reports*, 25(2):513-522.e3, 2018.
[2] J. Hughey and A. Butte. *Nucleic acids research*, 43 (12): e79, 2015.
[3] T. Hastie *et al. The Elements of Statistical Learning* (2nd Ed.), Springer, New York, New York, 2009.
[4] R. Bellman. *Dynamic programming*. Princeton University Press, 1975.
[5] L. Breiman. *Machine Learning*, 45 (1): 5âĂŞ32, 2001.
[6] J. Friedman. *The Annals of Statistics*, 29(5):1189-1232, 2001.
[7] T. Chen and C. Guestrin. Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Francisco, CA, 2016.
[8] T. Chen *et al. Package 'xgboost' Version 0.82.1*, CRAN Repository, 2019.

Mutation of synthetic constructs in *E. coli*

Duncan Ingram
Imperial College London
d.ingram@imperial.ac.uk

Mark Isalan
Imperial College London
m.isalan@imperial.ac.uk

Guy-Bart Stan
Imperial College London
g.stan@imperial.ac.uk

1 INTRODUCTION

When engineering *E. coli* cells to produce proteins of interest, a pertinent challenge is how to optimise the performance of the cell, whether that requires maximising protein expression, or sustaining a population over many generations. A key way to control cells' behaviour is through the design of synthetic constructs, and therefore being able to predict how construct composition affects host performance is of great interest. As the behaviour of both construct and host are mutually connected, controlling cell performance is inherently complex and can lead to unexpected outcomes such as circuit failure or stunted growth [2]. These phenomena are usually attributed to the over-consumption of shared cellular resources by heterologous gene expression [1], a phenomenon known as burden, and this needs to be carefully considered when designing cell systems.

In order to predict the variable effects of construct design in realistic cell environments, 'whole cell models' (WCMs) can be applied, which consider the interplay of key cellular processes given finite resources (figure 1). While existing WCMs are able to provide powerful descriptions of biological processes, there are many important phenomena that existing WCMs do not consider, and so their scopes are ultimately limited to idealised scenarios.

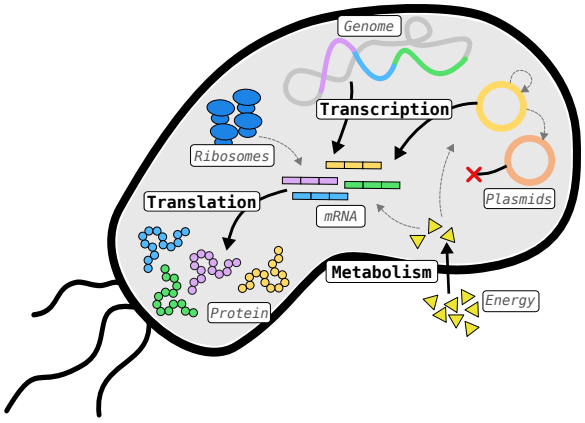


Figure 1: A simple 'whole cell model' (WCM), showing key processes in *E. coli*: metabolism, transcription and translation. Here we also consider the function of a synthetic plasmid (yellow circle), which produces its own proteins that causes a burden on the host. This construct can mutate during DNA replication (orange circle), alleviating the burden.

One such process that has yet to be explored in a WCM setting is the mutation of synthetic constructs. Copying errors during DNA replication often render plasmids obsolete [4], leading to a sudden drop in heterologous expression for that cell. While this is bad for our experiments, this mutation reduces the burden on the cell and allows it to grow faster, akin to positive feedback. Mutations to constructs are thus positively selected for, and the rate at which this occurs is important to understand and control if we want to optimise cell performance. Many factors affect mutation spread, and one significant contributor is the construct's genetic composition; depending on the construct's sequence, size and modular composition, its level of burden on the host will vary which in turn connects to mutation selection across the population [3]. To better understand these phenomena, therefore, requires modelling the spread of mutation in a WCM setting.

2 THE WHOLE-CELL MUTATION MODEL

To explore these phenomena, we first model a population of *E. coli* cells as a stochastic birth process, using engineered (E) and mutant (M) cells as two distinct states. E cells can divide to produce an M cell with a particular probability, while M cells always produce another M. In a chemostat, where cells are continually diluted to maintain a fixed number, we would expect that a population of E cells will gradually all mutate, with the rate at which this occurs being linked to burden.

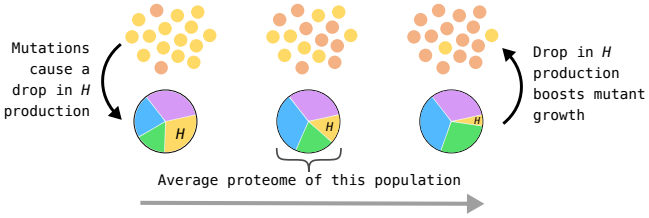


Figure 2: A population of E/M cells has a corresponding average fraction of different functional types of protein. We consider how these fractions change as the population mutates over time, given that the H fraction is directly linked to mutant growth rate.

To link our mutation model to burden, we consider different functional types of protein, for example ribosomal, enzymatic and heterologous. E cells produce heterologous (H) protein from their plasmids, so with more mutations, the

fraction of H in a population will drop. By linking the increase in mutants with the decline of the H fraction, we can model how construct burden affects the spread of mutation (figure 2).

We demonstrate the general effect of burden on mutation spread by starting with a chemostat of all E cells; as before, each cell contains different types of protein, including H. Over time, M cells are expected to take over the population, at a rate linked to the presence of burden (figure 3). If burden is ignored in our model (dashed lines), we find a slow decay of E cells, however when we link the H fraction to mutant cell growth, we can mimic the feedback that drives the positive selection of mutant cells. Alongside the rise of M cells is a decrease in H fraction, as expected.

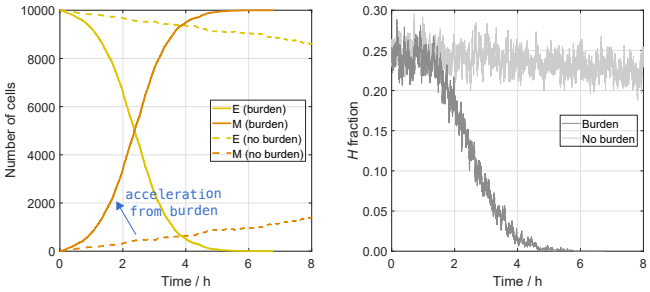


Figure 3: Left: the quantity of E and M cells in a chemostat over time, for cases where construct burden is ignored (dashed lines) and considered (solid lines). Right: the drop in H fraction over time, averaged over every cell.

This is a useful starting point to probe the way in which mutation spreads in a population of engineered cells, but further work will consider linking this to experimental data. For example, time-series data of the decay of different constructs (with different genetic components) can allow us to probe the specific mechanisms of mutation, as research by [4] provides. This can ultimately help us to design a model that predicts mutation spread given a particular construct design.

3 CONCLUSION

The mutation of synthetic constructs affects the behaviour of an engineered cell population over time. In order to understand and predict these effects, we require modelling the spread of mutation in the context of a realistic cell environment. To begin exploring these phenomena, we model cell mutation in a chemostat and link mutation rate to the cellular burden produced by heterologous gene expression. Our results highlight the positive selection of mutants in a population, and they act as a starting point to explore the specific mechanisms of burden. This will ultimately allow us to design synthetic constructs that deliver optimal cell performance with construct mutation in mind.

Bibliography

REFERENCES

[1] Borkowski, O., Ceroni, F., Stan, G.-B., and Ellis, T. Overloaded and stressed: whole-cell considerations for bacterial synthetic biology. *Current opinion in microbiology 33* (2016), 123–130.

[2] Brophy, J. A., and Voigt, C. A. Principles of genetic circuit design. *Nature methods 11*, 5 (2014), 508–520.

[3] Jack, B. R., Leonard, S. P., Mishler, D. M., Renda, B. A., Leon, D., SuaÌąrez, G. A., and Barrick, J. E. Predicting the genetic stability of engineered dna sequences with the efm calculator. *ACS synthetic biology 4*, 8 (2015), 939–943.

[4] Sleight, S. C., Bartley, B. A., Lieviant, J. A., and Sauro, H. M. Designing and engineering evolutionary robust genetic circuits. *Journal of biological engineering 4*, 1 (2010), 12.

Parallel Binary Sorting and Shifting with DNA

Tonglin Chen
chen5202@umn.edu
University of Minnesota
Minneapolis, MN

Marc Riedel
mriedel@umn.edu
University of Minnesota
Minneapolis, MN

1 INTRODUCTION

This abstract presents a novel scheme for sorting and shifting binary values stored in DNA. The computation is effected via the mechanism of toe-mediated strand displacement [5], [1]. Recent research by the DNA-computing community has demonstrated how data can be stored in DNA by "nicking" it using gene-editing techniques [2]. Nicking can expose toeholds for strand displacement. Wang et al. have proposed a scheme called SIMDNA, for "Single Instruction Multiple Data DNA," that implements parallel DNA strand displacement operations on binary bits [3]. In their scheme, binary bits are encoded in segments of DNA called "cells". Instructions consist of strand displacement operations on the encoded bits, changing the location of exposed toeholds. Building on the work of Wang et al., we propose a scheme for performing parallel sorting and shifting of binary bits.

2 DESIGN OF THE SYSTEM

The coding scheme that we use for binary values is shown in Figure 1. For each cell, there are five regions. Region 1 is always exposed as a toehold. Regions 2 through 5 are covered. When storing a bit value of 0, there is a nick between regions 2 and 3; when storing a bit value of 1, there is a nick between regions 4 and 5.

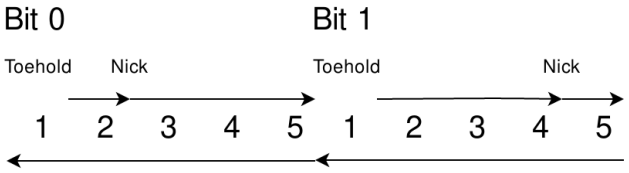


Figure 1: The coding scheme for bit values of 0 and 1.

Parallel Binary Bubble Sorting

We describe a scheme for sorting an arbitrary sequence of binary values. With the classical bubble sorting algorithm, a sequence of values is sorted by comparing and swapping adjacent values, from the beginning to the end of the sequence. With n values, the process is repeated n times, for a total run time of $O(n^2)$ steps. Sorting arbitrary values can be parallelized, but alternate passes are needed, sorting first odd and then even values, in order to avoid swapping conflicts.

With binary values, the bubble sorting algorithm is considerably simpler: it can be parallelized without alternate passes. To see why, note that there are four possible pairs of bits: $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$. Of these, only the pair $(1, 0)$ needs a swap. In any sequence of three bits, there can be at most one swap. Performing the compare and swap operations on all pairs of bits in sequence of n bits in parallel, the algorithm requires $O(n)$ parallel steps.

The basic idea for implementing the instructions for sorting with DNA strand displacement is first to identify all the $(1, 0)$ pairs and expose their corresponding toeholds. Then rewrite the data in those positions. Figure 2 shows the sequence of instructions for each step (a total of 12 instructions).

Instructions 1 and 2 identify the combination of $(1, 0)$. The toehold between a bit value of 1 and a bit value of 0 is replaced by a strand with a label of S1. Instruction 3 seals off the region exposed during Instruction 1 and 2. In instruction 4, the strand with label S1 is detached, exposing region 5, in the case of a bit value of 1, or region 2, in the case of a bit value of 1. In instruction 5, in the case of a bit value of 0, region 2 is temporarily covered by a strand with label S2. In instruction 6, a bit value of 1 is replaced by a strand with label S3 via the toehold at region 5. The strand is then detached and the bit value of 0 is written to the location of a bit value of 1 in instructions 7 and 8. In instruction 9, the temporary cover for a bit value of 0 is lifted. Then, in instructions 10 through 12, a bit value of 1 is written to the location of a bit value of 0 using the same scheme as instructions 6 through 8.

Left Shift Register

The scheme for sorting can readily be adapted to implement shifting of binary values. The basic idea for a left-shift register is the following:

- Find all the pairs $(0, 1)$ and $(1, 0)$
- Cover the toeholds for the pairs $(0, 0)$ and $(1, 1)$.
- Identify the pairs $(1, 0)$ and flip the bit values of 1.
- Identify the pairs $(0, 1)$ and flip the bit values of 0.
- Uncover all the toeholds for the pairs $(0, 0)$ and $(1, 1)$.

Our implementation the left shift register consists of 14 instructions per shift operation. The details are omitted due to space constraints.

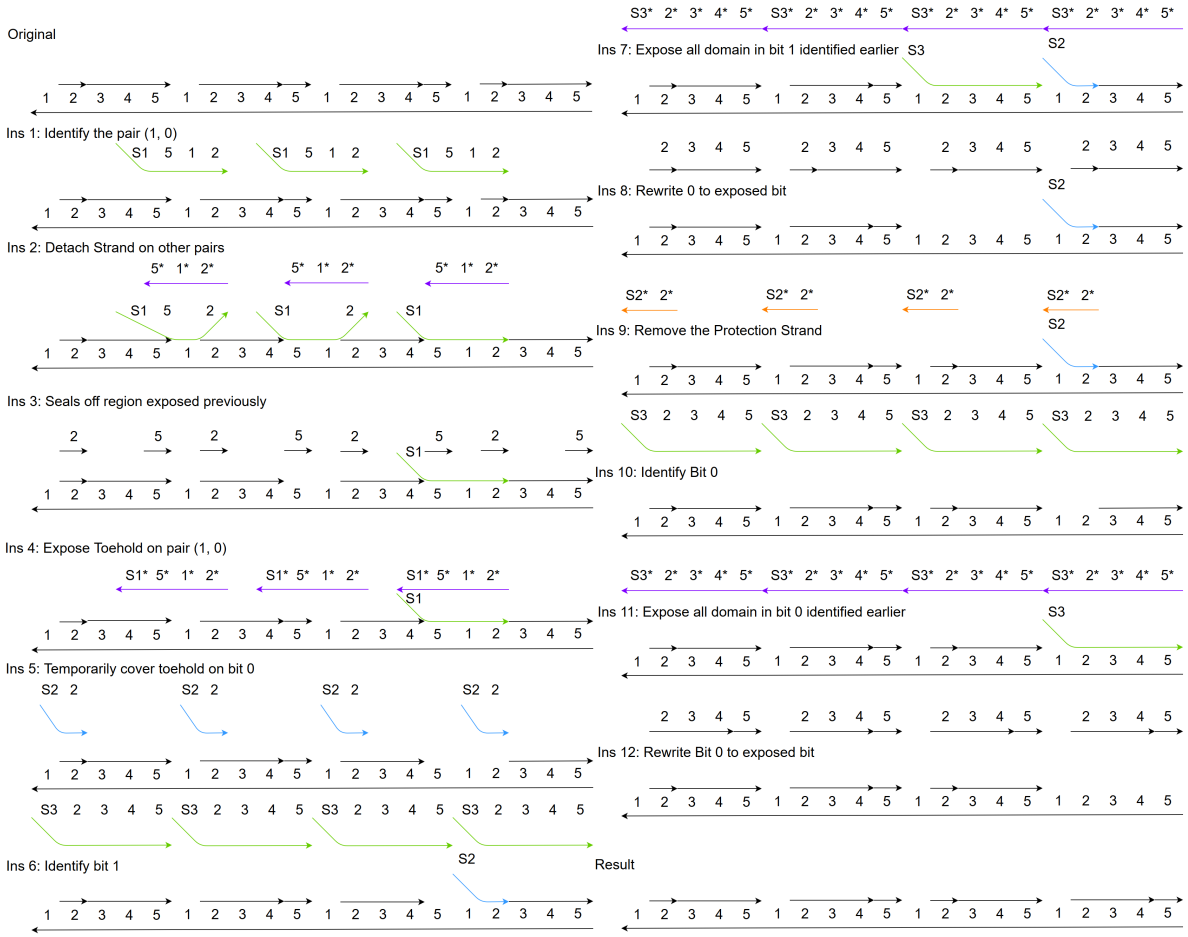


Figure 2: Instructions for one parallel step of binary bubble sorting. The original data encodes 0110. After one parallel step, the data encodes 0101.

3 DISCUSSION

We proposed a scheme for pairwise parallel execution of operations on binary values via strand displacement, and discussed two possible applications: sorting and shifting. We discussed the parallel binary bubble sorting algorithm in detail as it is a natural fit for the paradigm, since it only requires pairwise exchanges. We have only validated the scheme through simulation. A practical and experimental concern is the amount of "leakage" per operation: what fraction of strand displacement operations will not execute correctly. Each parallel step consists of about 10 separate instructions; leakage will compound over the course of these 10 operations. Using leakless strand displacement [4] as part of the design is a possible strategy to mitigate experimental error. In future research, we will explore alternate encoding schemes to minimize the number of instructions per step.

4 ACKNOWLEDGEMENT

We thank David Soloveichik, Cameron Chalk, and Boya Wang for helpful discussions. This work was funded by DARPA Grant W911NF-18-2-0032.

REFERENCES

[1] Soloveichik, D., Seelig, G., and Winfree, E. Dna as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences 107*, 12 (2010), 5393–5398.

[2] Tabatabaei, S. K., Wang, B., Athreya, N. B. M., Enghiad, B., Hernandez, A. G., Leburton, J.-P., Soloveichik, D., Zhao, H., and Milenkovic, O. Dna punch cards: Encoding data on native dna sequences via topological modifications. *Under Submission.*

[3] Wang, B., Chalk, C., and Soloveichik, D. Simdna: Single instruction, multiple data computation with dna strand displacement cascades. *Under Submission.*

[4] Wang, B., Thachuk, C., Ellington, A. D., Winfree, E., and Soloveichik, D. Effective design principles for leakless strand displacement systems. *Proceedings of the National Academy of Sciences 115*, 52 (2018), E12182–E12191.

[5] Yurke, B. A dna-fuelled molecular machine made of dna. *Nature 406*, 6796: 605 (2000).