ligify^{DB}: Extending the Biosensor Knowledgebase with Automated Annotation

Simon d'Oelsnitz* [1], Joshua D. Love[2],

Affiliations
[1] Synthetic Biology HIVE, Department of Systems Biology, Harvard Medical School, Boston, MA, 02115, USA
[2] Independent Web Developer. Bentonville, AR, 72712, USA
* To whom correspondence should be addressed: Simon d'Oelsnitz, simonsnitz@gmail.com

Abstract
Prokaryotic transcription factors (TFs) are indispensable small molecule biosensors with broad utility in biotechnology, but only a small fraction have been characterized. To address this issue, recent bioinformatic methods have leveraged information from enzyme reaction databases to predict effector molecules for a given transcription factor. Here, we extend this approach by systematically evaluating >10,000 small molecules in the RHEA enzyme reaction database for association to transcription factors. Ultimately, we generate a database of predicted associations for >3,000 unique transcription factors to >1,600 unique small molecules. We then create an interactive web application to access and query the database, which contains detailed information on small molecule structures, transcription factor structure and metadata, genome context, and prediction confidence metrics. The open-access database of predicted transcription factors presented herein aims to facilitate the systematic characterization and utility of genetic biosensors for chemical-control of biological systems.

SynBioKit: A Visualization and Validation Platform for Biological Designs

Metehan Unal
m.unal@keele.ac.uk
School of Computer Science and Mathematics, Keele University
Keele, Staffordshire, United Kingdom

Roberto Galizi
r.galizi@keele.ac.uk
School of Life Sciences, Keele University
Keele, Staffordshire, United Kingdom

Chris J. Myers
chris.myers@colorado.edu
Department of Electrical, Computer, and Energy Engineering, University of Colorado, Boulder
CO, United States

Göksel Mısırlı
g.misirli@keele.ac.uk
School of Computer Science and Mathematics, Keele University
Keele, Staffordshire, United Kingdom

1 INTRODUCTION

Synthetic Biology Open Language (SBOL) is a data standard developed to represent and share information about genetic circuit designs. The latest version of this standard (SBOL3), which has been adopted by the community, provides several advantages, such as enhanced expressive features to describe designs [5] and promote reusability and reproducibility. However, there is a lack of SBOL3-compliant tools. The utility of SBOL3 depends on the availability of modern tools to analyze, visualize, create, and validate genetic circuit designs. Without such tools, the adoption of this standard will not be truly achieved.

One way to improve the understanding of designs and their accessibility is to utilize efficient visualizations. In addition to the SBOL data model, SBOL Visual provides a comprehensive set of glyphs and symbols for different biological entities, such as promoters, terminators, and coding sequences [1]. Moreover, SBOL3 fully builds on semantic web technologies, paving the way to leverage already existing software tools and technologies, especially for graph-based approaches to store, visualize, and query data about biological designs. For example, the Genetta tool extracts relationships between gene products providing a higher-level protein-protein interaction networks for SBOL2 [3]. However, new tools are essential to utilize the featues of SBOL3. Moreover, SBOL designs include information about several different types of biological information and the complex relationships between them. As a result, designs need to be visualized using various abstractions and approaches.

Here, we present SynBioKit as a platform for visualizing synthetic biological designs, with validation and backward compatibility features being integrated. SynBioKit can visualize biological designs with different layout options, enforcing validations. As part of this work, we also developed a converter between SBOL2 and SBOL3 documents and vice versa. This feature is crucial to support existing tools and offer backward compatibility. We envisage SynBioKit to be a central platform for the synthetic biology community to analyze, visualize, and validate their designs using various options for different types of end users.

2 SYNBIOKIT PLATFORM

SynBioKit is an online platform that functions as a single-page application, featuring an editor to work with SBOL documents and convert them into various formats. It improves user experience by providing different visualization and conversion options, along with custom settings to control the resulting documents. For example, the "Design View" can visualize structural information such as the order of genetic parts, and the "Network View" shows the relationships of SBOL3 entities using different levels of abstraction.

The user interface is divided into two main areas. While the left side of the interface captures the user input, the right side displays the resulting outputs. These areas can be resized, allowing users to interact with SynBioKit more efficiently. Moreover, the visualization outputs can be downloaded as images.

One of the important features of SynBioKit is verifying that the SBOL documents comply with the validation rules. For example, when the user clicks the generate button to create visualizations, the validity of the design is checked.

Design View

In this view, the genetic circuit elements are presented in a backbone according to the SBOL Visual standard. This view builds on paraSBOLv [2], a Python library that can be programmed to create genetic circuit diagrams. The SynBioKit framework identifies the parent designs that act as containers and displays them. Figure 1 shows the order and types

of genetic parts within a design and the information about the template from which the design is inherited.

Network View

The network view has multiple visualization options such as "SBOL3 Graph", "Hierarchical", and "Gene Products".

The "SBOL Graph" view provides a detailed visual representation of an SBOL document (Figure 2). All top-level entities, such as components, sequences, and combinatorial derivations, as well as the child entities that connect these top-level entities and the relationships between all entities, are visualized as a graph. Users can interact with these graphs, for example, to focus on the neighborhood of specific design entities. They can also hide or show child entities and relationships using different coloring options to enhance the user experience. This view comes with various graph layout options, such as "dagre", "breadfirst", and "concentric" to analyze the flow of information, and "grid" and "circle" to focus on relationships among design entities.

The "Hierarchical" view provides a top-down view of genetic designs as a biological network (Figure 3). This view hides SBOL-specific child entities and focuses on the hierarchy between genetic designs and parts. Currently, all parent designs that act as containers are rendered within the same graph. The child parts are displayed in the order specified within each design, based on start and end locations and constraints such as "precedes".

The "Gene Products" view presents the interactions between small molecules, gene products, such as proteins, and other related entities. This view simplifies the visualizations further by providing additional useful abstraction.

Backward compatibility

SBOL3 introduces significant changes compared to SBOL2, including a simplified data model, improved modularity, and new entities. However, many existing tools and databases still rely on SBOL2. The development of a converter between SBOL2 and SBOL3 is essential to ensure continuity and accessibility within the synthetic biology community. In this study, we have also developed a tool to convert SBOL2 documents to SBOL3 and vice versa, utilizing the libSBOLj3 library [6] for SBOL3 and libSBOLj [8] for SBOL2. SBOL2 files from the "SBOL Test Suite" were converted to SBOL3 and back with a 95% success rate (180 out of 189). A few files could not be converted due to errors in the original files. This converter enables the use of legacy datasets with the latest tools, and older platforms can support a subset of the new data model.

3 METHODS

SynBioKit has a multi-layered architecture with frontend and backend components, which contain additional layers to provide a modular, scalable, and efficient framework. The backend consists of web services. A Java web service utilizes the libSBOLj3 [6] library to perform detailed validation and SBOL-specific operations. It creates graphs ready to be rendered in the form of JSON documents. A Python web service is used to create design views, building on paraSBOLv [2]. It also utilizes and maps SBOL Visual glyphs [7] and the matplotlib library to enhance rendering details. A React-based and interactive user interface renders visualizations that are processed by the backend components. Graph-based visualizations are created using Cytoscape.js [4]. A Node.js middleware orchestrates the communication, involving API requests and data transfers, between the frontend and the backend components.

4 FUTURE WORK

SynBioKit will be extended in collaboration with the synthetic biology community to provide a range of visualizations that can be helpful for different types of end users. Currently, the converter is a standalone library and does not support the conversion of measurement and design-build-test entities. When the converter is finalized, it will be directly integrated into SynBioKit for seamless SBOL2 support.

ACKNOWLEDGMENT

We thank the SBOL community and the HARMONY 2025 participants. M.U., R.G. and G.M. were supported by the BBSRC grant BB/Z517367/1. For the purposes of open access, the authors have applied a Creative Commons Attribution (CC-BY) license to any Accepted Author Manuscript version arising from this submission.

REFERENCES

[1] Baig, H., et al. Synthetic biology open language visual (sbol visual) version 3.0. *Journal of Integrative Bioinformatics 18*, 3 (2021), 20210013.

[2] Clark, C. J., Scott-Brown, J., and Gorochowski, T. E. parasbolv: a foundation for standard-compliant genetic design visualization tools. *Synthetic Biology 6*, 1 (2021), ysab022.

[3] Crowther, M., et al. Genetta: a network-based tool for the analysis of complex genetic designs. *ACS Synthetic Biology 12*, 12 (2023), 3766–3770.

[4] Franz, M., et al. Cytoscape. js: a graph theory library for visualisation and analysis. *Bioinformatics 32*, 2 (2016), 309–311.

[5] McLaughlin, J. A., et al. The synthetic biology open language (sbol) version 3: simplified data exchange for bioengineering. *Frontiers in Bioengineering and Biotechnology 8* (2020), 1009.

[6] Misirli, G. libsbolj3: a graph-based library for design and data exchange in synthetic biology. *Bioinformatics 39*, 8 (2023), btad525.

[7] Misirli, G., Beal, J., Gorochowski, T. E., Stan, G.-B., Wipat, A., and Myers, C. J. Sbol visual 2 ontology. *ACS Synthetic Biology 9*, 4 (2020), 972–977.

[8] Zhang, Z., et al. libsbolj 2.0: a java library to support sbol 2.0. *IEEE life sciences letters 1*, 4 (2016), 34–37.

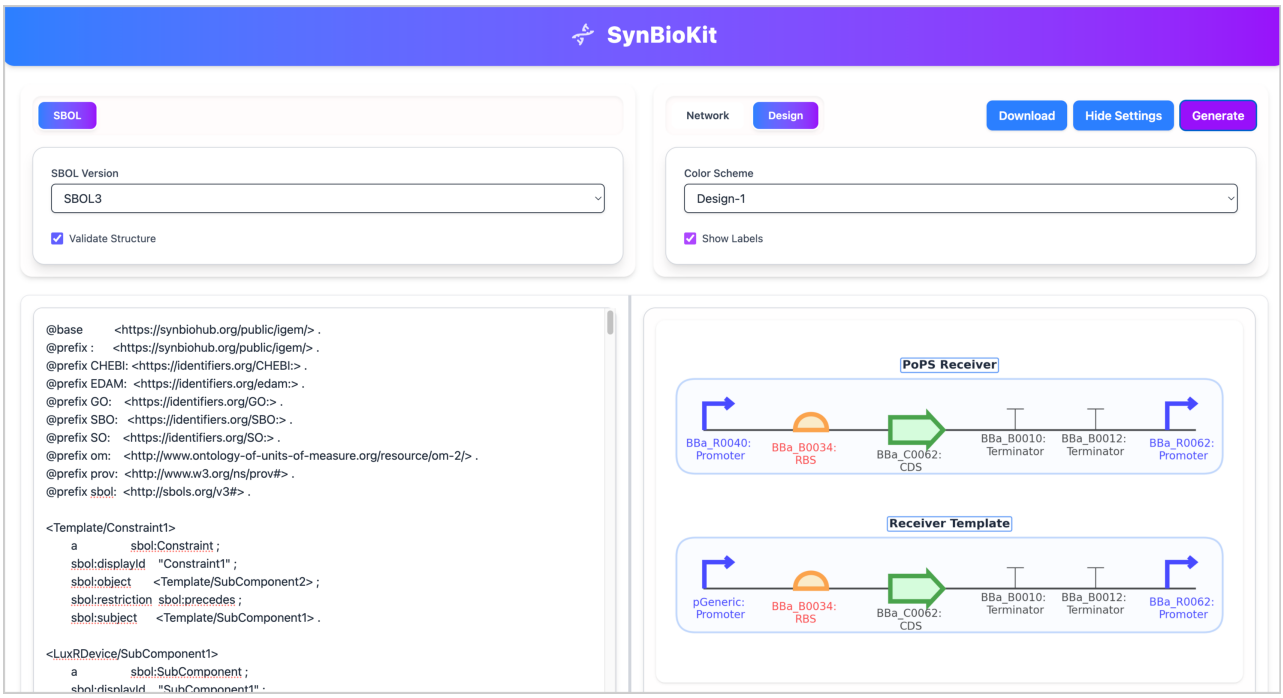


Figure 1: SynBioKit editor view. The editor is divided into two sections to take descriptions of designs and visualize them. The "PoPS Receiver" example design is visualized using SBOL Visual glyphs and paraSBOLv. The design inherits from a template design and replaces the generic promoter with a custom promoter.

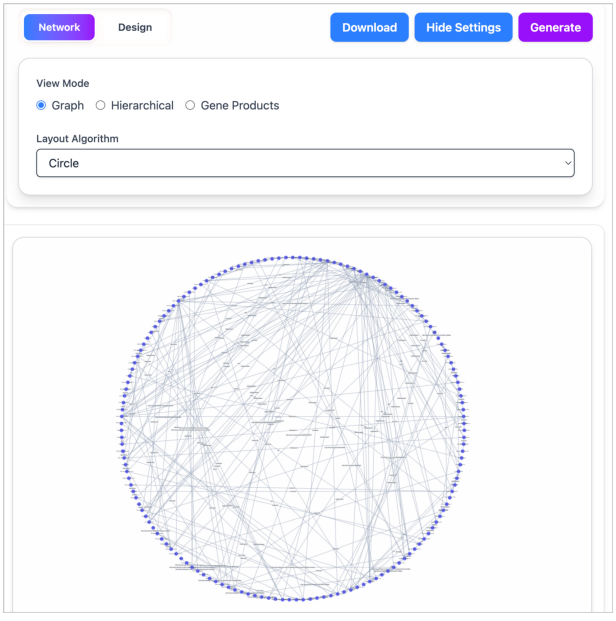


Figure 2: An example of a graph view. The designs from Figure 1 are visualized using the circle layout. The entities can be examined via the zoom function, and other visualization types offer a simplified view to enhance understanding.

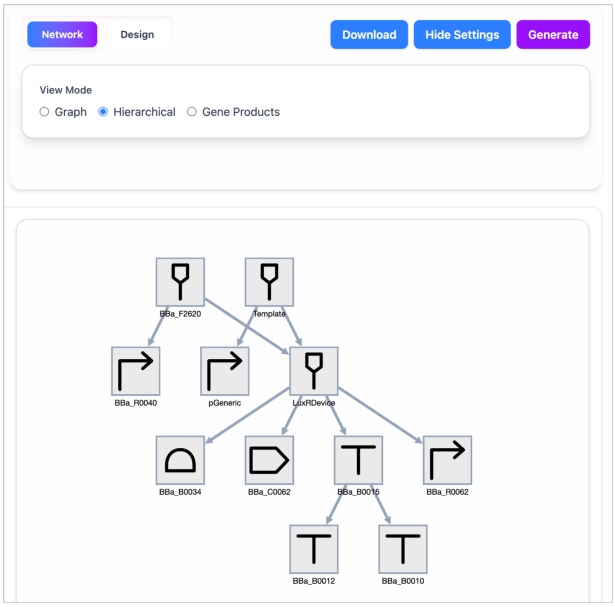


Figure 3: An example of a network view. The designs from Figure 1 are shown hierarchically as a graph. Subcomponents are ordered within each design.

Rule-based generation of synthetic genetic circuits - recent progress in v 2.0 -

Masayuki Yamamura*
Ryoji Sekine
Institute of Science Tokyo
Tokyo, Japan
my@c.titech.ac.jp
ryoji_722@msn.com

Kazuteru Miyazaki
NationalInstitutionfor AcademicDegrees
andQualityEnhancementof HigherEducation
Tokyo, Japan
teru@niad.ac.jp

Naoki Kodama
Tokyo University of Science
Tokyo, Japan
nkodama@rs.tus.ac.jp

Daisuke Kiga*
kiga@waseda.jp
Waseda University
Tokyo, Japan

1 INTRODUCTION

To design cellular behavior based on the synthetic biology approach, the appropriate selection of network motifs consisting of genetic parts, as well as the parts in a motif, is essential. In contrast to manual design, design based on logic programming allows the extensive generation of combinations of parts[1]. Model parameterization and numerical calculation are also required to evaluate the cellular behavior specifications. An automation tool designed for Boolean networks assigns biological parts to nodes within these networks and determines those parts' dynamic behavior to verify the designs' feasibility[2]. Tools calculating other types of networks are also developed[3][4], and databases for the models or simulation results are provided[5][6].

These four years, we have developed an automation tool for genetic circuit design by using Prolog inference engine. Table1 shows a brief summary of the improvement history of our system. Our IWBDA 2022 abstract referred to our previous paper, which describes a synthetic genetic circuit for reprogramming and diversifying the gene-expression status of living cells[7]. In the manual design procedure in the paper, we initially combined the toggle switch and gene overexpression motifs in a cell because a manual phase-space analysis of the toggle switch with and without the overexpression shows bifurcation between bistability and monostability, and both are required for the reprogramming and diversification process. For design automation of the circuit using an inference engine, we developed a combination of Inside Prolog and C++ codes available in Zenodo[8].

This version, we call v1.0, implements full function into one software system. Reading the configuration files for genetic circuit design rules and the required specification, our program will enumerate candidates of a network design with a combination of reaction parameters. It can produce an explanation of the inference process by a proof tree. The numerical simulation module equipped a cache mechanism for simulation results. We found Prolog inference engine has enough flexibility to connect with external numerical simulations, and also has enough scalability to extend a program by just adding new inference rules.

Using such flexibility and scalability of the Prolog inference engine, our other modification of the toggle switch in the IWBDA 2023 code was the addition of intercellular communication, which governs autonomous cell-type diversification on an epigenetic landscape showing bifurcation from monostability to bistability. The diversification of cell states on the epigenetic landscape was the specification for an automated design of the cell-population behavior regulated by cell-cell communication. This diversification has been achieved by using our synthetic circuit in living cells[9]. In the inference-engine-based design flows for the communication-dependent cell-type diversification, parameterized models initially designed for multistability were modified to incorporate cell-cell communication, and the modified model was examined for the specification of cell state movements on the epigenetic landscape.

This v1.1 achieves a parallel execution of inference rules for multi cellular system with cell-cell communication. Based on a server/client model, we added explicit parallel inference primitives in the inference rule. Inference rules can communicate with each other by inter-thread communications. We extended the cache function of numerical simulation results and the pretty print function for proof trees.

In IWBDA 2024, we designed a multi-layer system consisting of a receiver cell and two sender cells, each of which can produce its unique cell-cell communication molecule. The inner states of the two sender cells are inputs for a logic gate in the receiver cell. After an incubation of each sender

*All authors were supported by JST, CREST Grant Number JPMJCR21N4, Japan

cell in liquid medium for a specific time, each supernatant containing the communication molecule is transferred to the other liquid culture of the receiver cell. The two inner states of the sender cells thus become an input pattern for the logic gate. One communication molecule activates GFP production, and the other activates the production of a repressor for GFP production. With an adequate parameter set, only one of the four input patterns produces GFP efficiently.

We call this version as v1.2, which is a complete form of parallelism. We extended parallel design rules by adding request forms to other processes. In order to optimize parallel calculation in given computational environment, uses can specify the group of parallel computation for each parallel execution, and also users can specify a certain server to be assigned. A minor version up v1.3 was done to accept requests to upload a design rules file and download a result file through Web-based protocols.

2 RESULTS AND CONCLUSION

After IWBDA 2024, we achieved major version up to v2.0, which realizes a generalization on motif representation. We assumed a simple typical formula format for a specific chemical reaction for v1s. We extended to allow reaction formula which includes multiple productive terms in following three types: (1) multiplicative, (2) additive and (3) enzymatic form.

$$\alpha \cdot \frac{(\frac{x_1}{K_1})^{n_1}}{1+(\frac{x_1}{K_1})^{n_1}} \cdot \frac{(\frac{x_2}{K_2})^{n_2}}{1+(\frac{x_2}{K_2})^{n_2}} \cdots \tag{1}$$

$$\alpha \cdot \frac{(\frac{x_1}{K_1})^{n_1} + \frac{x_2}{K_2})^{n_2} + \cdots}{1+(\frac{x_1}{K_1})^{n_1} + \frac{x_2}{K_2})^{n_2} + \cdots} \tag{2}$$

$$\alpha \cdot (\frac{x_1}{K_1})^{n_1} \cdot (\frac{x_2}{K_2})^{n_2} \cdots \tag{3}$$

We restricted the maximum number or genes to 20, and the maximum number of productive terms to 10. We changed the interface specification for numerical simulation programs according to the general representation. We also prepared several primitive functions to improve the computational performance for a numerical simulation. For example, users can set the grid size for the initial value to search steady states, and can select growth models like logistic curves other than exponential ones assuming low cell density.

For v2.0, we have tested our system by implementing the mutual repression toggle switch with communication molecules, which is the test bed for v1.3, by new representation. Then we examined new design examples for logic gate systems whose description requires this update of the design tool. Figure1 shows (A) the circuit design, (B) the truth table and (C) a schematic diagram for a typical time series expected in numerical simulations.

Our Prolog-based code generates genetic networks that satisfy a given specification by combining rules that define network motifs, assumed parameters for motif components, and mathematical conditions derived from the specification. The logical structure of rules can help us uncover alternative methods of genetic manipulations. Prolog's flexibility allows the inclusion of alternative methods from multiple programmers, enhancing the scale and complexity of the generated networks. Beyond v2.0, we will extend the application of our system for any Synthetic Biology genetic circuit design.

This project was motivated by a criticism upon black-box AI approaches with high performance but no explanation such as Deep Learning for genetic circuit design in Synthetic Biology. We have achieved a white-box approach to satisfy given specifications with logical inference and numerical simulation. In the development process, we found Large Language Models have strong affinity with logical inference rules. Please remark that we can check evidences for generated inference rules and mathematical formula quite scientifically, in contrast with Humanities area applications of generative AI. One of the further issues would be automatic generation of initial network designs, mathematical formula and new inference rules. We expect that the combination of logical inference and LLMs become a breakthrough for the design issues in Synthetic Biology.

REFERENCES

[1] Pederson, M. and Phillips, A. Towards programming languages for genetic engineering of living cells. J. R. Soc. Interface 2009, 6S437–S450.

[2] Jones, T. S.; Oliveira, S. M. D.; Myers, C. J.; Voigt, C. A.; Densmore, D. Genetic circuit design automation with Cello 2.0. Nat. Protoc. 2022, 17, 1097-1113.

[3] Tas, H.; Grozinger, L.; Goni-Moreno, A.; de Lorenzo, V. Automated design and implementation of a NOR gate in Pseudomonas putida. Synth. Biol. (Oxf) 2021, 6, ysab024.

[4] Boada, Y.; Reynoso-Meza, G.; Pico, J.; Vignoni, A. Multi-objective optimization framework to obtain model-based guidelines for tuning biological synthetic devices: an adaptive network case. BMC Syst. Biol. 2016, 10, 27-0.

[5] McLaughlin, J. A.; Myers, C. J.; Zundel, Z.; Misirli, G.; Zhang, M.; Ofiteru, I. D.; Goni-Moreno, A.; Wipat, A. SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology. ACS Synth. Biol. 2018, 7, 682-688.

[6] Yanez Feliu, G.; Earle Gomez, B.; Codoceo Berrocal, V.; Munoz Silva, M.; Nunez, I. N.; Matute, T. F.; Arce Medina, A.; Vidal, G.; Vitalis, C.; Dahlin, J.; Federici, F.; Rudge, T. J. Flapjack: Data Management and Analysis for Genetic Circuit Characterization. ACS Synth. Biol. 2021, 10, 183-191.

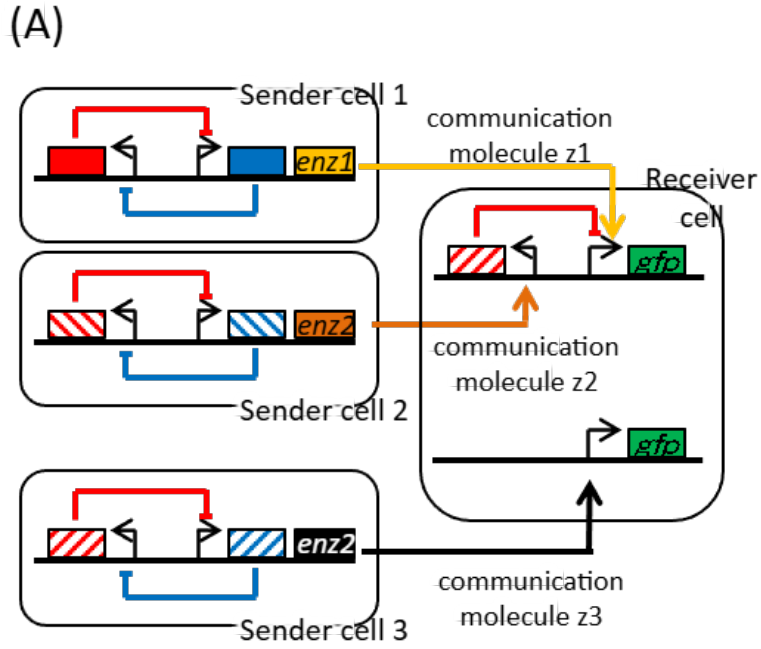
[7] Ishimatsu, K.; Hata, T.; Mochizuki, A.; Sekine, R.; Yamamura, M.; Kiga, D. General applicability of synthetic gene-overexpression for cell-type ratio control via reprogramming. ACS Synth. Biol. 2014, 3, 638-644.

[8] https://doi.org/10.5281/zenodo.8148662.

[9] Sekine, R.; Yamamura, M.; Ayukawa, S.; Ishimatsu, K.; Akama, S.; Takinoue, M.; Hagiya, M.; Kiga, D. Tunable synthetic phenotypic diversification on Waddington's landscape through autonomous signaling. Proc. Natl. Acad. Sci. U. S. A. 2011, 108, 17969-17973.

Table 1: A summary of the system improvement

ver	date	progress	external simulator	testbed circuit
1.0	2023.9	total system starts - network -> param set - proof tree pretty pr - cache sim results		mutual repression toggle switch
1.1	2023.12	parallelize (thread comm) - server/client model - parallel pretty print - parallel caching	precision improvement - convergence test - equilibria search	toggle switch (parallelized)
1.2	2024.6	parallelize (process comm) - divide process - assign server	add functions - SingleCellTimeCourse - configure initial values	toggle switch (with comm chem) multilayer logic gate (AND only)
1.3	2024.12	remote up/down-load - via web		
2.0	2025.6	expand representation - new interface to simulator	four formula types - multiplicative - additive - enzymatic	toggle switch (new form) multilayer logic gate (universal)



(B)

Sender cell 1	Sender cell 2	Sender cell 3	Receiver cell
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

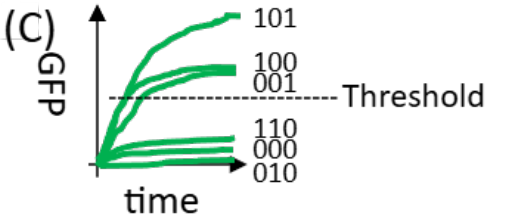


Figure 1: Genetic circuits designed by the inference engine based on Prolog

(A) A multilayer genetic system composed of three sender cells and one receiver cell implementing a logic gate. Each sender cell is capable of synthesizing its distinct cell-cell communication molecule. The internal states of the sender cells serve as inputs to the logic gate embedded in the receiver cell. After individual incubation of the sender cells in liquid culture for a defined period, their supernatants—potentially containing communication molecules—are transferred into the culture medium of the receiver cell.

(B) The truth table and the ideal time course of GFP expression in the receiver cell. The implemented logic gate follows the expression ($z1$ AND (NOT $z2$)) OR $z3$. Communication molecule $z1$ activates GFP expression, whereas $z2$ induces a repressor that inhibits GFP production from a $z1$-regulated hybrid promoter. Molecule $z3$ independently drives GFP expression from a separate promoter. Under appropriately tuned parameters, the receiver cell produces GFP levels that exceed the threshold required for logic evaluation.

Patterned Polyacrylamide Gel as a Testbed for Molecular Communication

Manao Ito, Yutaka Hori*
{manao110,yhori}@keio.jp
Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522, Japan

1 INTRODUCTION

Molecular communication (MC) between cells realizes more complicated functions than those achievable by a single cell operating in isolation. In nature, for example, cells form biofilms to protect themselves [4], and our immune system has highly complicated functions [2] as a result of MC. Similarly, synthetic MC systems also enable advanced functionalities through cell-to-cell information exchange via molecular communication. Since MC is primarily driven by passive diffusion of signal molecules, it is essential to understand how the dynamics of the communicating agents are affected by diffusion related parameters such as diffusion coefficient and diffusion length. Therefore, it is desirable to develop a testbed that includes features such as (1) compartmentalization of sender and receiver systems, (2) easily adjustable diffusion parameters, and (3) high versatility of modules such as signal molecules.

To test the effect of diffusion parameters on MC systems, several testbeds were developed that allow systematic variation of spatial arrangements. One approach uses patterning of *E. coli* colonies on LB agar plates [1], allowing for high versatility of the choice of signal molecules. Another approach employs wells carved on the HEMA-EDMA gel [3], where the pore size allows only the diffusion of signal molecules and prevents the diffusion of *E. coli*. However, the former suffers from maintaining the spatial configurations due to the growth of the colony on the well, while the latter is constrained by the limited compatibility with hydrophobic signal molecules due to their adsorption to the gel surface.

To address these limitations, we propose a gel-based device composed of hydrophilic polyacrylamide that was fabricated using a silicone mold designed via CAD and produced with a 3D printer. The proposed device allows high versatility of the choice of signal molecules due to its hydrophilic property, while maintaining a well-defined spatial arrangement suitable for analyzing molecular communication dynamics over time. As a demonstration example, we performed a communication experiment between sender and receiver *E. coli* strains with varying distances to evaluate the effect of communication length on system dynamics.

This work was supported in part by JSPS KAKENHI Grant Number JP23H00506.

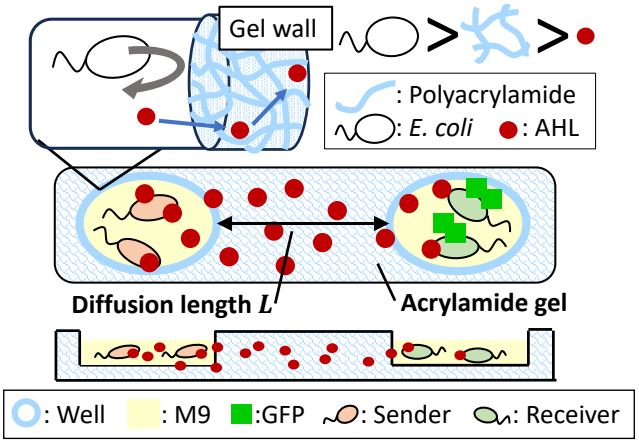


Figure 1: Overview of the gel-based MC testbed

2 MECHANISM OF THE PROPOSED DEVICE

The proposed device is a polyacrylamide gel-based platform designed to test MC between spatially separated cell populations (Fig. 1). The device comprises an array of wells molded into polyacrylamide gel using a silicone mold designed with CAD software and fabricated with a 3D printer, which enables control over the spatial arrangement of sender and receiver regions. A key feature of the device is its ability to physically confine *E. coli* cells within individual wells while allowing the free diffusion of signal molecules such as N-acyl homoserine lactones (AHL). Compartmentalization of *E. coli* and diffusion of AHL are realized due to the pore size of the polyacrylamide gel, which is approximately from 10 nm to 200 nm. The diffusion coefficient D can be tuned by total concentration of monomer acrylamide and weight percentage of crosslinker BIS as the pore size is controlled by polyacrylamide chains linked by BIS in polyacrylamide gel. The diffusion length L can be determined by the geometric configuration of the wells in the mold, which can be defined using the CAD software. In addition, the hydrophilic nature of polyacrylamide minimizes nonspecific adsorption of signal molecules to the gel matrix, making the device both practically accessible and biologically compatible.

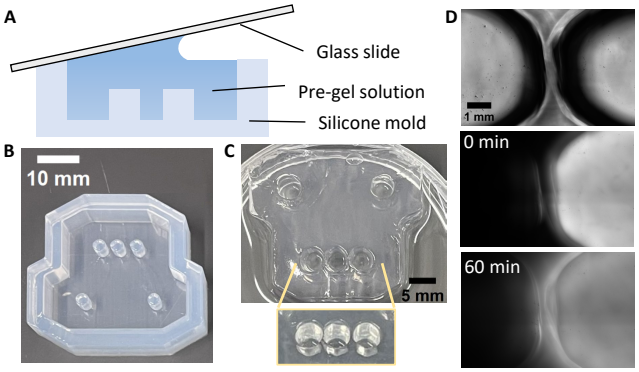


Figure 2: Mold and replica of polyacrylamide gel-based device. (A): Overview the gel device fabrication. (B): 3D printed silicone mold. (C): Polyacrylamide gel-based device (D): Time-lapse images of fluorescein diffusion.

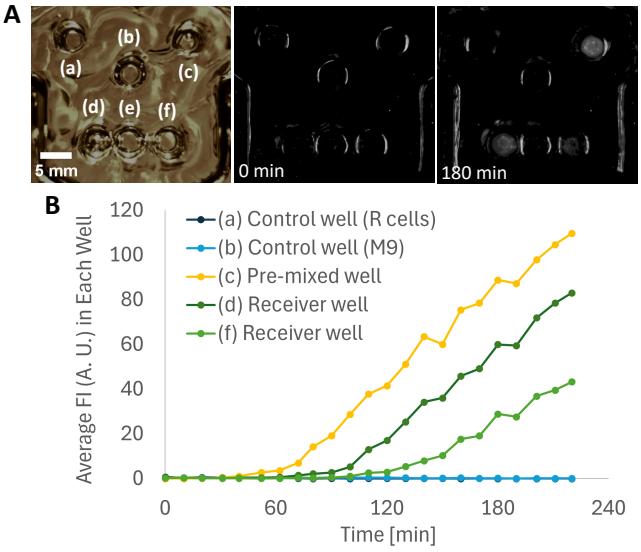


Figure 3: Demonstration of *E. coli* based MC system. (A) MC system built on the device. The role and the content of each well is as follows: (a) Control well, R cells only, (b) Control well, M9 media only, (c) Pre-mixed well, R cells and 50 nM AHL, (e) Sender well, 50 nM AHL, (d) Receiver well ($L_d \approx 0.4$ mm), R cells, (f) Receiver well ($L_f \approx 1.5$ mm), R cells. (B) Average fluorescence intensity in the each well.

3 RESULTS AND DISCUSSION

Fabrication and verification of the gel device

A silicone mold was formed by a 3D printer (SD-01, HOTTY POLYMER) using liquid silicone rubber. The mold was filled with pregel solution, then covered with a glass slide to prevent oxygen and incubated for gelation (Fig. 2**A, B, C**). The pregel solution contains 10 w/v % of 29:1 acrylamide-BIS (06141-35, Nacalai tesque), 0.67% of 10 w/v % ammonium peroxodisulfate (06284-04, Nacalai tesque) and 0.1% N,N,N',N'-tetramethylethylenediamine (TEMED) (33401-72, Nacalai tesque). Mold printing takes 3 to 5 hours depending on the designed pattern, and gelation is completed in a few hours, and thus, the overall fabrication process can be finished within a day.

To test the capability of the fabricated gel-based device as a testbed of MC, we next verified its diffusion capacity. As a test molecule, we used fluorescein, which has a higher molecular weight than AHL and therefore is expected to diffuse more slowly according to Einstein-Stokes equation. Despite its relatively slow diffusion, diffusion of fluorescein from the right well to the left well was clearly observed (Fig. 2**D**). These results confirm that the proposed device has sufficient diffusion capacity to operate MC systems.

Demonstration of E. coli based MC system

Finally, to demonstrate that MC systems can be tested with the proposed device, we built an MC setup using AHL solution and the receiver *E. coli* (R cells) as shown in Fig. 3**A**. GFP fluorescence increased only in the pre-mixed well (c) and in the receiver wells (d and f) (Fig. 3**B**), reflecting that AHL mediated MC successfully activated downstream GFP expression. In particular, a longer diffusion distance resulted in a greater delay in the response as observed in wells (c), (d), and (f), implying that the difference in response time due to diffusion kinetics was successfully captured using the proposed framework. More quantitatively, the final FI value of well (f) was 52% of that of (d). The time T when FI started to be detected in them was $T_d \approx 60$ min and $T_f \approx 90$ min, respectively. These results demonstrate that the device is suitable as a testbed for analyzing the spatiotemporal kinetics of MC systems.

4 CONCLUSION

We have developed a gel-based testbed for MC systems. The proposed device allows easy adjustment of the diffusion parameters and has a high versatility of signal molecules. The device can be used to analyze the dynamics of MC and contribute to understanding and constructing MC systems.

REFERENCES

[1] Doong, J., et al. Length and time scales of cell-cell signaling circuits in agar. *bioRxiv* (2017), 220244.
[2] Hwang, I. Cell-cell communication via extracellular membrane vesicles and its role in the immune response. *Molecules and cells 36*, 2 (2013), 105–111.
[3] Vaiana, C. A., et al. Characterizing chemical signaling between engineered "microbial sentinels" in porous microplates. *Molecular Systems Biology 18*, 3 (2022), e10785.
[4] Wang, Y., et al. Biofilm formation and inhibition mediated by bacterial quorum sensing. *Applied Microbiology and Biotechnology 106*, 19 (2022), 6365–6381.

SeqTrainer: Encoding Synthetic Biology Data for Machine Learning

Sai Wong
sai.wong@sjsu.edu
San Jose State University
San Jose, California

Chris Myers
Chris.Myers@colorado.edu
University of Colorado Boulder
Boulder, Colorado

Gonzalo Vidal
Gonzalo.VidalPena@colorado.edu
University of Colorado Boulder
Boulder, Colorado

1 INTRODUCTION

Synthetic biology aims to engineer biological systems, with its priority being sequence-to-expression modeling. With the rapid rise of machine learning (ML), many avenues in synthetic biology have been opened. ML allows for high-throughput analysis of genetic components, uncovering complex patterns that traditional methods do not capture. Techniques such as deep learning have shown great merit in modeling promoter strength, gene regulation, and protein expression from raw sequence data. Moreover, the use of large language models (LLMs) has proven to be effective in automating laboratory tasks through the use of artificial intelligence (AI) agents. However, the effectiveness of ML models depends heavily on the availability of data. As the field continues to grow, connecting experimental biology with computational tools like ML models is becoming increasingly essential for accelerating innovation and scientific discoveries.

The Synthetic Biology Open Language (SBOL) [2] standard was created to represent biological designs systematically, and SynBioHub [4] serves as a repository of these designs. SynBioHub is the database with the largest amount of standardized biological designs encoded in SBOL and therefore is a great place to obtain a large amount of training data for ML models. Classical ML models can be trained on experimental values such as RNA expression values and encoded DNA sequences, which can be easily encoded in SBOL and uploaded to SynBioHub.

However, because the data in SynBioHub is encoded in a triple-store XML format, these data values cannot be directly passed into ML models, which expect numeric features. Thus, if biological researchers wish to use the data in SynBioHub, they face the task of having to encode this data, which becomes infeasible without some experience in ML. As a result, SynBioHub currently lacks accessible, preprocessed datasets, which limits researchers' ability to efficiently develop models trained on the data from SynBioHub.

Our tool, SeqTrainer, aims to help researchers efficiently collect the data they need to train models by preprocessing the data stored in SynBioHub. We developed a Python package that streamlines querying and preprocessing data from SynBioHub, generating features for ML models. By integrating SBOL data querying, feature engineering (including k-mers, PWM, and GC skew), and graph neural network (GNN) modeling, this project will help researchers to efficiently analyze synthetic constructs and generate predictions for their data.

2 METHODOLOGY

This section presents our implementations and the pipelines that our Python package provides.

The Python package first allows the user to search for the parts or experiments they want encoded, whether or not this data was uploaded themselves. When data is encoded in SBOL format, the constituent parts and properties can be easily obtained. In pySBOL2 [1], various methods are given to directly interact with the SynBioHub API, allowing the user to upload their created components or pull from the database. Our package seamlessly integrates with pySBOL2; Users can pull their data from SynBioHub, and the package takes that as input. With a simple SPARQL query, we can extract the necessary components for our first module of data preprocessing.

In our first module, when the user inputs their data into our package, there can be many properties and values that are important, so it becomes difficult to filter out which properties are most important for training the model. To remedy this, we allow the user to provide a configuration file where the user can specify which DNA sequences they would like extracted and which labels from the input data they would like to use as their label in their model (to predict). Once the user provides this configuration, we can convert all of their data into an organized, tabular format by selecting the properties necessary from the SBOL encoding using a query. Our package mainly handles DNA sequence pre-processing; we use several R and Python scripts created by Urtecho et al. [5] for one-hot encoding, extracting PWM scores, calculating GC scores, and generating k-mers (Figure 1). We provide these preprocessing methods to the users through the Python package, which allows the data to finally be converted into a numeric format, suitable for training.

For our second module, rather than producing data in a format used by classical ML models such as RandomForest

and LogisticRegression, we produce data suitable for training GNNs. However, there are constraints upon what the user can input into the package. In ML, each of the training data must be in the same format (dimensions must be the same). Especially for training a GNN for tasks such as graph regression, each of these graphs must be in the same format. Thus, for this module, our package expects to receive different graphs (SBOL files) that hold the same structure, which is a simple task by querying SynBioHub for multiple parts in the same experiment. For each of these SBOL files stored as an XML file, we first convert these files into Resource Description Framework (RDF) with N-Triples serialization. We use the tool AutoRDF2GML [3], which expects RDF files, and use the tool to create both topological and content-based features from our graph. As this tool requires a user configuration file to determine which nodes and edges from the graph should be encoded, we allow the user to provide information such as the types of nodes they want encoded, while auto-generating the rest of the configuration based on this information. Using the CSV outputs from the tool, we manipulate this data by mapping each file to a node or edge, and encode it in PyTorch Geometric's HeteroData. Labels such as for regression and classification tasks can be appended to this HeteroData using our method which queries each file for the label. Using this data, users can train their own GNNs from PyTorch Geometric or use our provided model which performs graph level regression.

3 RESULTS

To test our package, we began by using data gathered by Urtecho et al.[5], when they performed experiments for promoter discovery in *E. coli*. We mapped out a graph structure represented, which can easily be represented in SBOL format (Figure 2). With this, we encode each row in the CSV which provides details such as the expression levels of a promoter, the sequence itself, the location, etc in SBOL, resulting in multiple SBOL graphs/files for each promoter tested in their data. Moreover, we utilize the experimental information obtained from the paper such as the media used to create our SBOL representations. Initially, we tested our first module by specifying the features we needed (DNA sequence and expression level) in the configuration and successfully extracted the SBOL data back into a CSV or tabular format. We then follow the pipeline by using our package's preprocessing methods and then training RandomForest and LogisticRegression models to observe very similar results to Urtecho et al. during their analysis, demonstrating our package's capability to go from SBOL to a trained model similar to recent publications.

Afterwards, we tested out our second module for training a GNN. Using the graph structure that was mapped out previously, we could easily create the configuration file needed for AutoRDF2GML[3]; with our aforementioned pipeline, our package could then take that file and encode each of our SBOL files into HeteroData objects with their corresponding expression values. With this, we complete the pipeline by using our provided GNN model suited for graph level regression to train a GNN model which predicts expression values from a sample design in this experiment encoded in SBOL. Due to a lack of processing power, our module could not train on a lot of data points; we eventually plan to evaluate our results and compare them to other ML models.

4 CONCLUSION

As ML becomes a greater focus in synthetic biology, tools that bridge the gap between biological standards like SBOL and ML frameworks are becoming increasingly important. Our tool SeqTrainer addresses this need by providing a pipeline for querying, preprocessing, encoding and training ML models on data formated in SBOL stored in SynBioHub. By offering compatibility with classical ML and GNN-based approaches, our tool enables researchers to easily leverage their data for predictive tasks. In the future, we hope to expand our project by providing more flexibility in terms of what the user can input into our packages and implementing training workflows using high performance computing (HPC). We also want to explore the use of this architecture for the prediction of a promoter sequence that would drive certain expression level. We hope SeqTrainer will benefit the community by allowing researchers to perform scalable, data-driven design and analysis in the intersection between synthetic biology and ML.

REFERENCES

[1] Bartley, B. A., McLaughlin, J. A., Beal, J., Nguyen, T., Clancy, G. M., Roehner, N., et al. pysbol: A python package for genetic design automation and standardization. *ACS Synthetic Biology 8*, 7 (2018).

[2] Buecherl, L., Beal, J., McLaughlin, J. A., Bartley, B. A., et al. Synthetic biology open language (sbol) version 3.1.0. *Journal of Integrative Bioinformatics 20*, 1 (2023).

[3] Färber, M., Lamprecht, D., and Susanti, Y. Autordf2gml: Facilitating rdf integration in graph machine learning. *arXiv preprint arXiv:2407.18735* (2024). Accessed Jul. 31, 2025.

[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 (2018), 682–688.

[5] Urtecho, G., Bittner, L. R., Cocanougher, J. R., and Tabor, J. J. Genome-wide functional characterization of *escherichia coli* promoters and sequence elements encoding their regulation. *eLife 12* (2023).

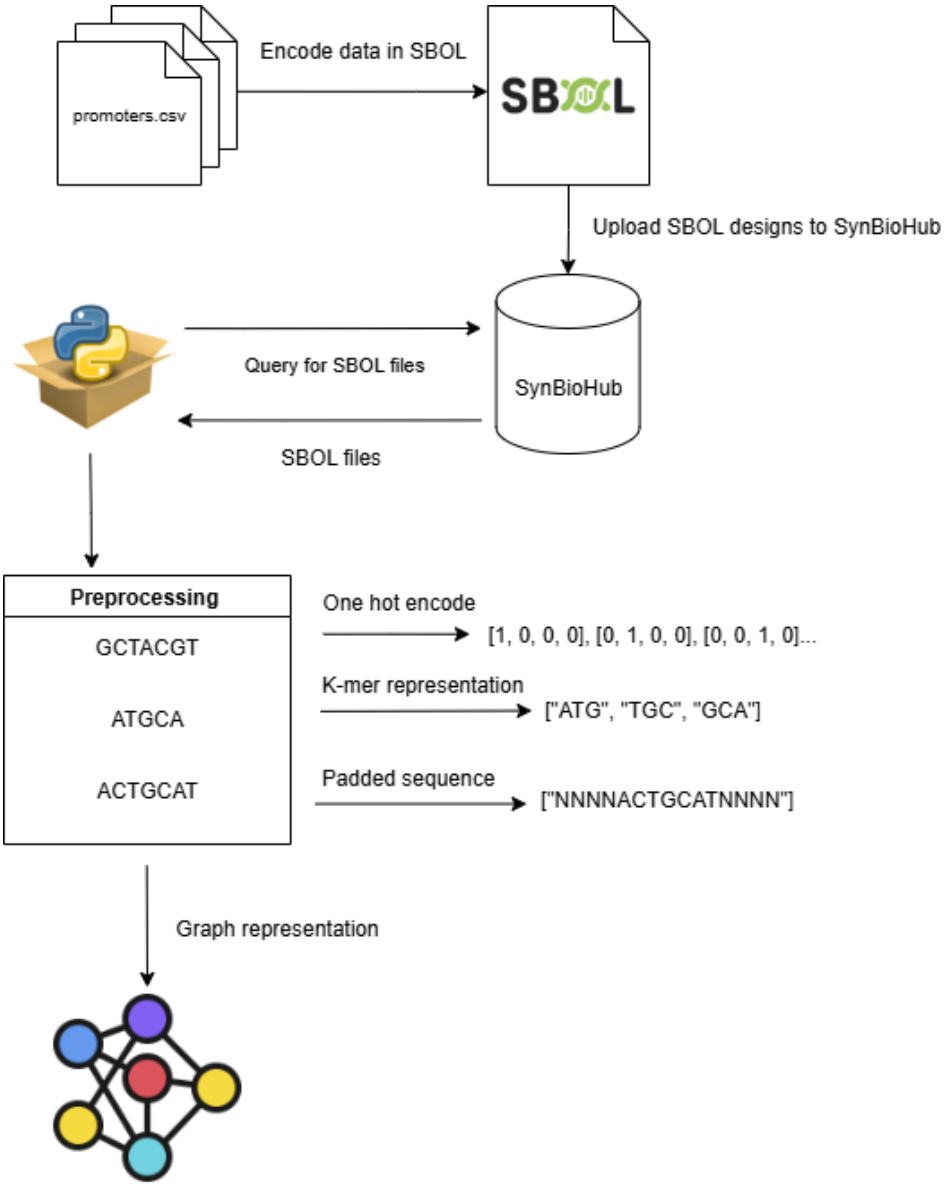


Figure 1: Diagram of the encoding pipeline. Data in CSV format is first converted into SBOL and uploaded to SynBioHub. Experimental data can be queried to SynBioHub and downloaded as SBOL files. It then provides different preprocessing and encoding methods to get the final vectors for training ML models on tabular and graph representations.

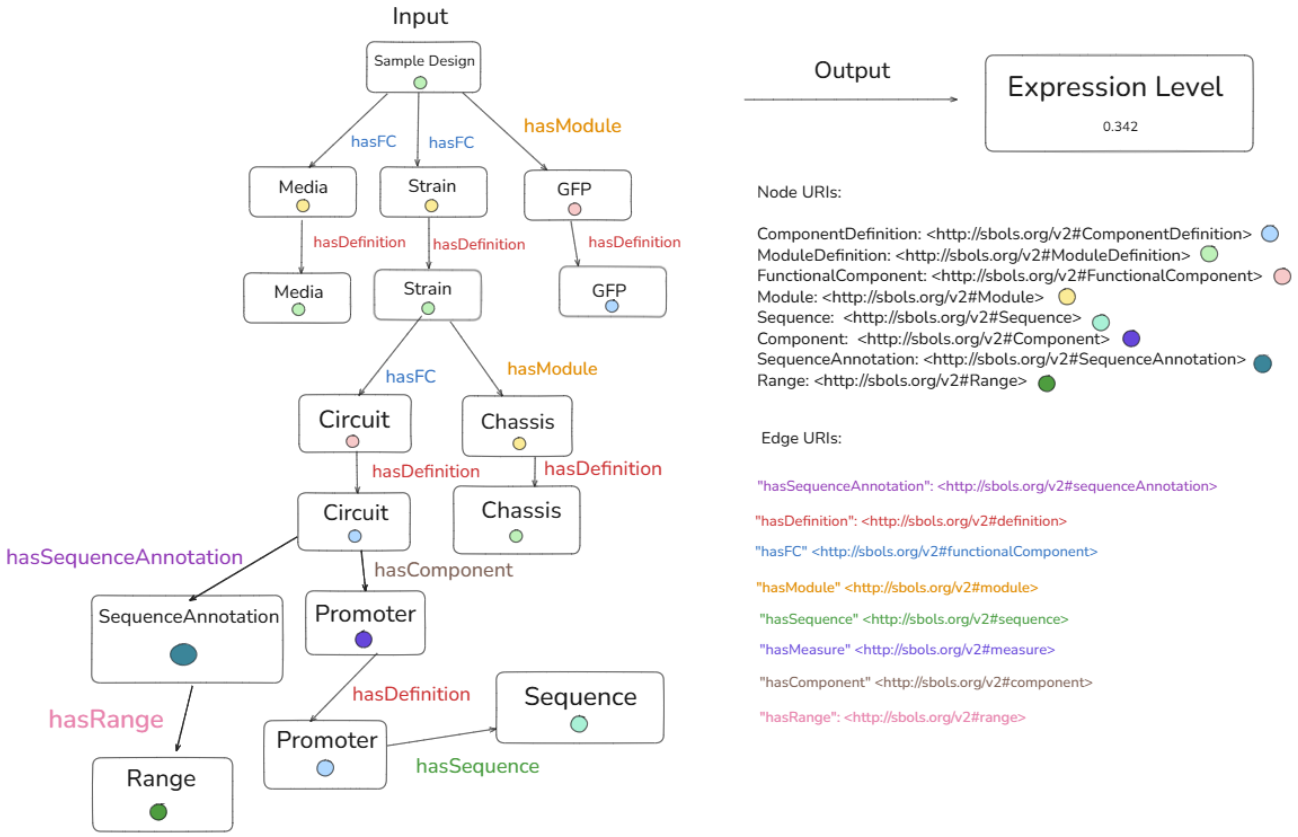


Figure 2: Diagram of input and output for the GNN model. The input for the GNN is a sample with media and a strain that produces GFP. The strain is created by transforming a chassis with a genetic circuit. The genetic circuits the is composed by a promoter which has the DNA sequence. The output is a label with an expression level measure. For our SBOL data, we mapped out a graphical visualization for the inputs and output of our GNN. This graph represents one SBOL file that AutoRDF2GML converts to vectors using the configuration file, which requires the nodes and edges URIs as shown.

Dr. Plant, a Plants' Disease Detection Model using Machine Learning

1st Ahmad Nahar Quttoum
Department of Computer Engineering
Faculty of Engineering
The Hashemite University
Zarqa 13133, Jordan

2nd Bashar Alshboul
Department of Computer Engineering
Faculty of Engineering
The Hashemite University
Zarqa 13133, Jordan

3rd Hamza Assad
Department of Computer Engineering
Faculty of Engineering
The Hashemite University
Zarqa 13133, Jordan

4th Shatha Almarashdi
Department of Computer Engineering
Faculty of Engineering
The Hashemite University
Zarqa 13133, Jordan

5th Wa'ed Azimee
Department of Computer Engineering
Faculty of Engineering
The Hashemite University
Zarqa 13133, Jordan

***Abstract*—In recent years, artificial intelligence (AI) has witnessed remarkable advancements, leading to its integration across diverse sectors including agriculture, healthcare, and environmental monitoring. This work presents the development of an intelligent, AI-based plant disease detection system built upon the VGG16 convolutional neural network architecture. The system is designed to classify plant leaves as either healthy or diseased, and in the case of disease identification, it provides the disease name, underlying cause, and tailored prevention and treatment recommendations. At the core of the system lies a robust image recognition engine powered by transfer learning and fine-tuning, enabling the model to fine-tune its parameters for high classification accuracy even with limited agricultural datasets. To bridge the gap between advanced technology and end-users in rural settings, the system features a highly intuitive and user-friendly web interface specifically designed for farmers, many of whom may have limited technical knowledge. This ensures that the tool is not only powerful but also accessible and practical for real-world farming use. Beyond its practical application in precision agriculture and sustainable farming, this proposal allows hands-on experience in cutting-edge AI domains such as deep learning, transfer learning, and fine-tuning. Early evaluations indicate that the VGG16-powered system offers high accuracy, fast processing, and significant potential to revolutionize plant health monitoring in resource-constrained environments. This innovation aligns with the global push towards smart agriculture, contributing to more resilient and informed farming communities.**

***Index Terms*—Artificial Intelligence, VGG16, Plant Disease, Smart Agriculture, Crop Health Management.**

I. Introduction and Problem Statement

The agricultural sector is a fundamental pillar of the global economy, contributing approximately 4% to the global GDP and providing employment for approximately 1 billion people worldwide. However, plant diseases pose a significant threat to global food security, responsible for up to 40% of annual crop losses, amounting to over $220 billion in economic damage. Traditional methods of disease detection, such as visual inspections by agricultural experts, are time-consuming and prone to errors. This challenge is particularly pronounced for smallholder farmers in developing countries, who often lack access to modern diagnostic tools. As a result, there is delayed disease identification, which increases dependence on harmful pesticides. These practices not only raise production costs but also contribute to environmental degradation [1], [2].

In Jordan, agriculture contributes approximately 5.9% to the national GDP and employs around 2.4% of the workforce. The sector is vital for rural livelihoods, with numerous communities relying on it for their income and sustenance. However, Jordan faces several challenges, the most pressing of which is severe water scarcity, ranking among the most water-scarce nations globally. Agriculture in Jordan consumes over 85% of the available fresh water resources, a situation further exacerbated by the impacts of climate change and growing population pressures. This overuse of water resources threatens the long-term sustainability of agricultural practices and the country's food security [3].

A. Contribution

The contribution of this work comes from the development of an AI-powered plant disease detection system that uses deep learning models, specifically the VGG16 convolutional neural network (CNN), to accurately classify plant diseases based on leaf images. The system aims to provide farmers with a cost-effective, efficient, and scalable solution for the diagnosis of plant disease in real time, eliminating the need for expert intervention. By integrating the system into a web-based platform, it will be easily accessible, particularly to farmers in rural and remote areas. According to the Food and Agriculture Organization (FAO), plant diseases significantly undermine global food security by reducing crop yields and agricultural production each year [**?**]. Using this approach, the project allows faster and more accurate disease identification, supports proactive crop management, and contributes to the advancement of precision agriculture and global food security.

Hence, with the proposed Dr. Plant model, we aim to overcome the following challenges:

- Late Disease Detection: Plant diseases, when identified late, can lead to significant crop losses. Timely detection is essential to prevent further spread and reduce damage.
- Lack of Access to Expert Knowledge: Farmers often lack the necessary access to agricultural experts capable of diagnosing plant diseases accurately and promptly.
- Pesticide Overuse: Without accurate disease detection, farmers tend to overuse pesticides, which contributes to environmental pollution and escalates production costs.

II. The Model

During the preliminary phase, comprehensive research was conducted across several data repositories and platforms, including GitHub, Hugging Face, and Kaggle to identify suitable plant disease image datasets. The primary objective was to locate a dataset that combines high image quality with relevance to widely cultivated plant species.

A. Model Selection and Evaluation

To determine the optimal deep learning architecture for plant disease classification, multiple CNN-based and transformer-based models were implemented, trained, and evaluated on the Plant-village dataset under controlled conditions. Key performance metrics such as accuracy, precision, recall, and confusion matrices were considered, alongside factors like model complexity, training time, and deployment feasibility.

The architectures explored includ:

- VGG16: Delivered the best balance between classification accuracy, training stability, and inference efficiency. This made it the most suitable model for integration into the planned web-based system.
- ResNet-50: It showed promising performance, yet it required a number of training epochs, produced inconsistent noisy results complicating convergence and reliability.
- Custom-built CNN: Exhibited poor overall performance, suffering from high overfitting and noisy outputs, limiting its effectiveness for reliable disease classification despite being computationally efficient.
- Vision Transformer (ViT): Demonstrated unstable performance with significant variability in outcomes across different runs. Due to reaching a dead-end in optimization and instability, further work on ViT was discontinued.

Based on these experimental outcomes, VGG16 was selected as the final model for implementation due to its superior balance of accuracy, robustness, and computational efficiency.

1) Justification for VGG16 Model Selection: : Out of the multiple models evaluated (VGG16, ResNet-50, custom CNN, ViT), VGG16 was selected as the core classification model based on the following merits:

- Balanced Performance: VGG16 achieved high accuracy, precision, and recall across multiple disease classes while maintaining inference efficiency.
- Computational Simplicity: Its relatively lower architectural complexity makes it well suited for real-time classification in a web-based system.
- Transfer Learning Compatibility: It showed strong adaptability when fine-tuned on the PlantVillage dataset, reducing training time and improving generalization.
- Stability: It exhibited consistent convergence behavior and minimal overfitting across multiple training sessions.

B. Training and Validation

In this work, we utilized the Plant Disease Dataset sourced from Kaggle, which originally contained over 87,000 high-resolution RGB images (JPEG format) representing plant leaves affected by various diseases as well as healthy samples. The dataset covers 38 distinct classes, each representing a specific combination of plant species and disease type, including a category for healthy leaves. All images were resized to ensure compatibility with the VGG16 model input requirements.

To further enhance the model's ability to generalize and handle real-world variability, data augmentation techniques were applied during training. These included random rotations, shifts, zooming, brightness adjustments, shearing, and horizontal flipping. The purpose of augmentation was to synthetically increase the diversity of training images, allowing the model to become more robust to different plant leaf orientations, lighting conditions, and natural variances ultimately reducing overfitting and improving performance on unseen data.

III. Results and Discussion

To access the performance of our proposed model, several key evaluation metrics were used throughout, such metrics as: complexity, training times, overfitting risks, and accuracy which served as the main metric during training and validation to measure the overall proportion of correct predictions. Figures 1, 2, and 3 present a detailed comparison for the performance metrics of three alternatives: Custom CNN, ResNet50, and the chosen model VGG16. Fig. 4 summarizes our findings in terms of suitability.

IV. Conclusions

This work explored the implementation of AI-powered plant disease detection using deep learning models, specifically focusing on VGG16. The proposed Dr. Plant model offers a powerful solution to detect plant diseases efficiently while minimizing pesticide use and environmental impact. The integration of such AI-based model into agricultural practices has the potential to revolutionize the way farmers monitor plant health, offering a low-cost, scalable, and accessible tool to enhance agricultural productivity and sustainability.

References

[1] Food and Agriculture Organization (FAO). (2021). "Cooperation at all levels and funding critical for plant health and food security, FAO says."

[2] Wolfert, S., Ge, L., Verdouw, C., and Bogaardt, M.-J. (2017). Big Data in Smart Farming – A review. Agricultural Systems, 153, 69–80. https://doi.org/10.1016/j.agsy.2017.01.023

[3] Al-Tal, R. M. (2025). How agriculture is powering Jordan's economy. The Jordan Times. Retrieved from https://jordantimes.com/opinion/raad-mahmoud-al-tal/how-agriculture-powering-jordans-economy

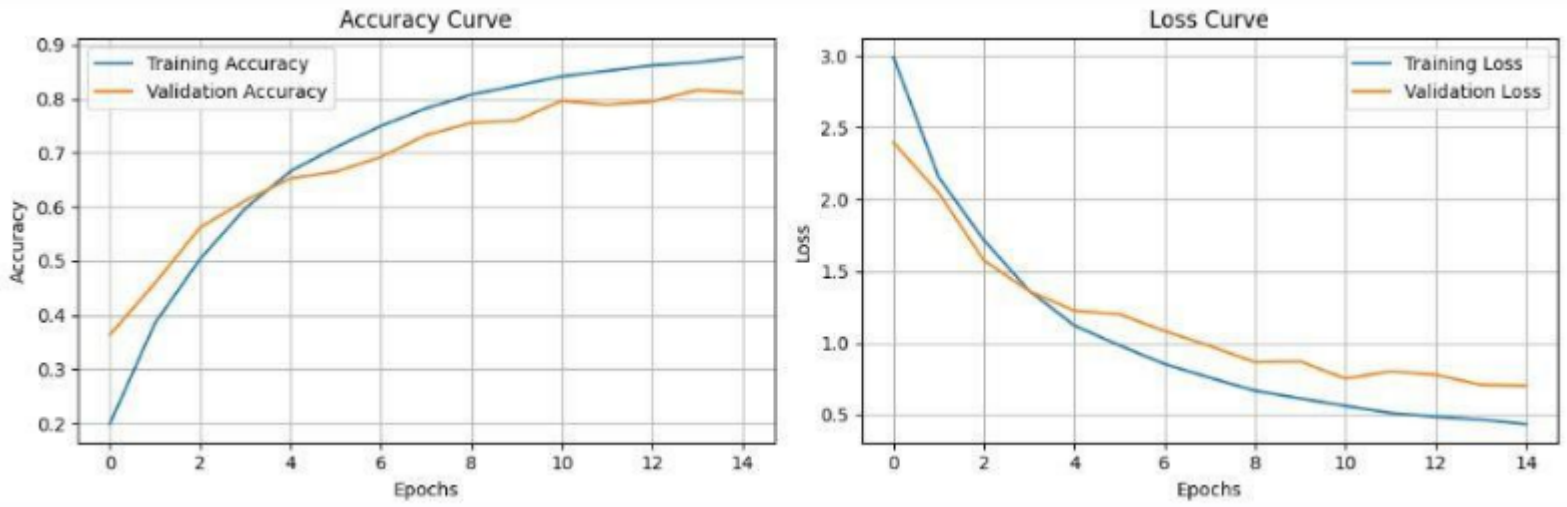


Fig. 1. Figure 3. Learning curves of Custom CNN

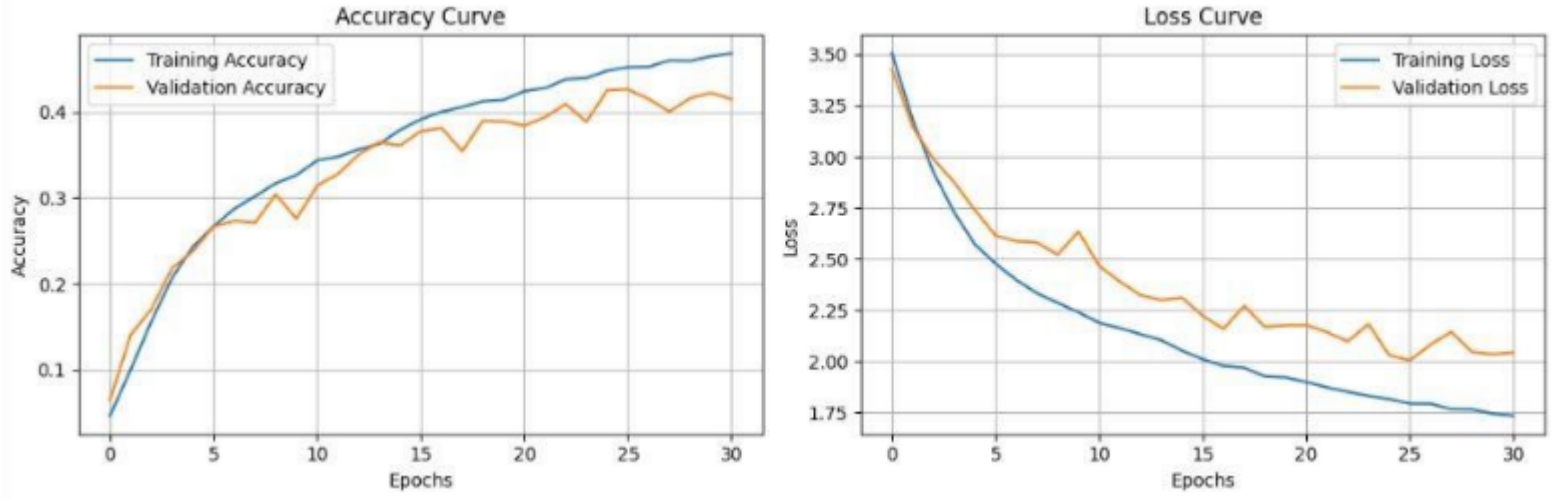


Fig. 2. Learning curves of ResNet50

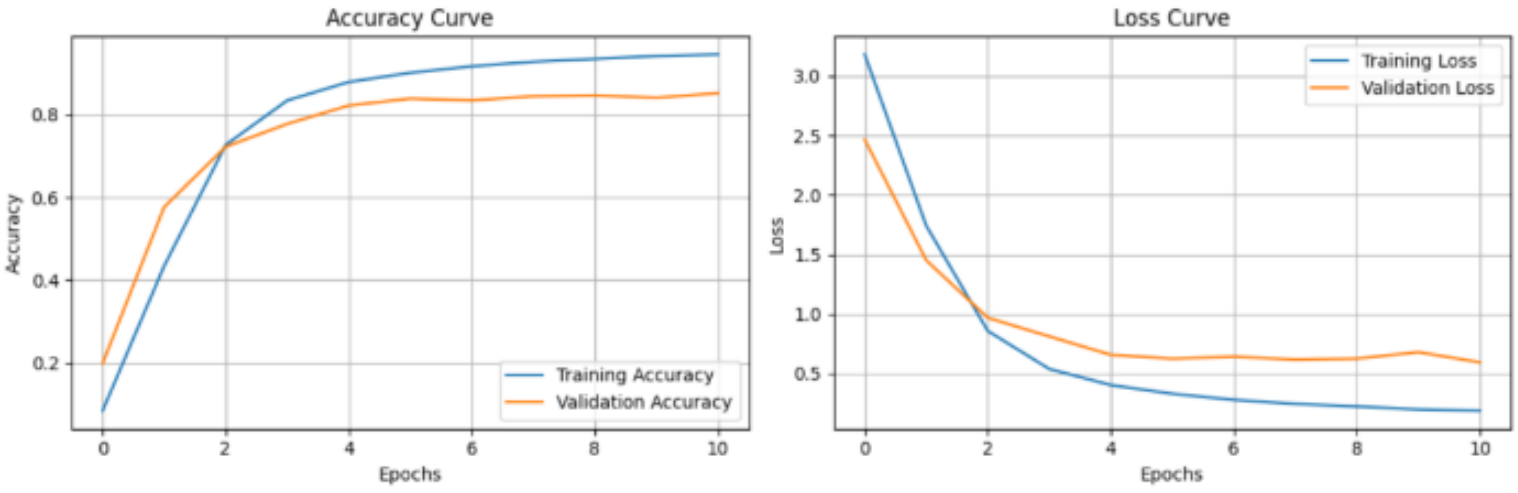


Fig. 3. Learning curves of VGG16

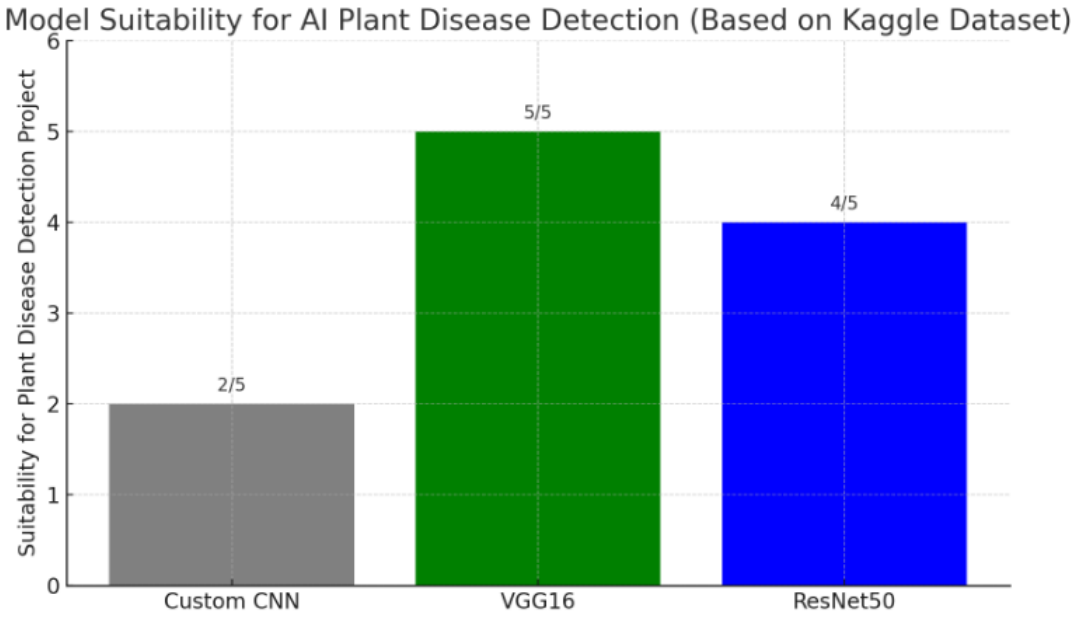


Fig. 4. Models' suitability findings

Degradation-Driven Failure Minimization in Genetic Circuits Through Model Checking

Landon Taylor[1], **Zhen Zhang**[1], **Lukas Buecherl**[2]
[1]Electrical and Computer Engineering, Utah State University, [2] Biological Engineering, Utah State University
lukas.buecherl@usu.edu

The genetic toggle switch is a foundational motif in synthetic biology, enabling bistable behavior in which a cell can maintain one of two distinct stable states in response to transient inputs [1]. In a common implementation, the toggle switch is composed of two transcriptional units: one in which TetR represses LacI expression, and another in which LacI represses the expression of TetR and YFP. The states are toggled by aTc and IPTG, which bind TetR and LacI, respectively, effectively removing the repressors. Once the toggle switch reaches a given state, its long-term stability depends on various system parameters, including protein production and degradation rates, which govern the dynamics of state maintenance and transitions.

In this work, we focus on how the degradation rates of the toggle switch proteins, namely LacI, TetR, and YFP, influence the probability of unintended state reversal after the system reaches a steady state. Degradation is a particularly attractive design parameter because it can be modulated through engineered degradation tags, offering a practical route for tuning circuit performance.

METHODOLOGY

We modeled the toggle switch system as two *continuous-time Markov chain* (CTMC) models, implemented in the PRISM modeling language and analyzed using the PRISM probabilistic model checker [2]. The toggle switch model include LacI, TetR, and YFP species, each with production and degradation reactions. Two separate models were created to represent the circuit initialized in the "on" (YFP high) and "off" (YFP low) states.

We systematically varied the degradation rates (k_d) of YFP, LacI, and TetR across nine values, [0.00375, 0.005, 0.00625, 0.0075, 0.00875, 0.01, 0.01125, 0.0125, 0.015], chosen to be evenly spaced around iBioSim's default k_d of 0.0075 [3]. With nine possible values for each of the three rates, each model was evaluated over $9^3 = 729$ unique parameter combinations. For each configuration, a Python script updated the degradation rates in the PRISM model, and PRISM's CTMC transient analysis calculated the probability that YFP would cross an undesired threshold within 2100 seconds (approximately one cell cycle), indicating a state switch. CTMC transient analysis was chosen because it yields *exact* probabilities, unlike stochastic simulations, which provide only approximations.

The resulting output was processed using a Python script to identify the parameter combinations yielding the *lowest* probability of failure in each model. Because each degradation rate influences the circuit's behavior differently, its effects on the YFP high and YFP low models are not the same. Thus, we use a set of metrics to weigh the probability results from both models and determine a balanced and optimal configuration. These optimal configurations suggest how experimentalists might tune protein degradation rates via engineered degradation tags to enhance circuit robustness.

RESULTS

Our preliminary analysis reveals that each degradation rate has a unique impact on the failure probabilities. Figure 1 is a violin plot, with the plot width representing the density of data points per probability. This shows that the YFP high model experiences a much larger probability variation over all combinations of parameters, while the YFP low model experiences a generally-low failure probability.

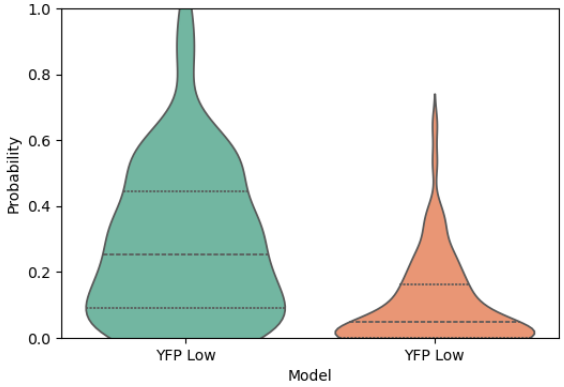


Figure 1: Distribution of Probability of Failure

The degradation rate of LacI is negatively correlated to the probability of failure in YFP high, but it is positively correlated to the average probability of failure in YFP low, as shown in Figure 2. Similarly, increasing the degradation rate for TetR increases failure probability in YFP high but decreases it in YFP low (see Figure 3), and an increased degradation rate for YFP has little impact on YFP low but greatly increases the probability of failure for YFP high (see Figure 4).

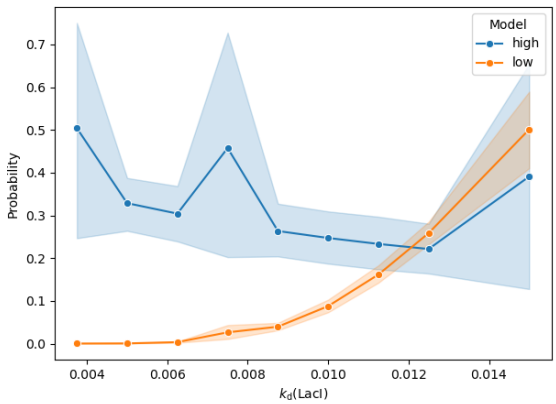


Figure 2: Average Probability of Failure vs kd of LacI

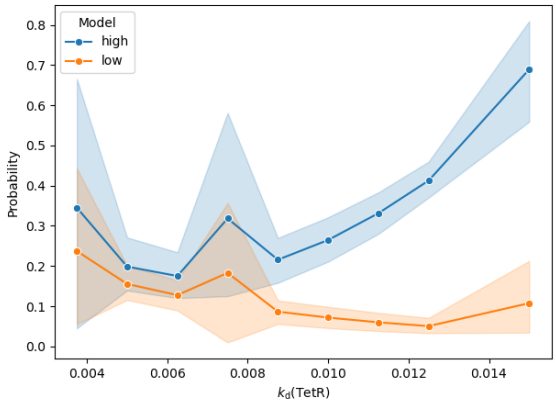


Figure 3: Average Probability of Failure vs kd of TetR

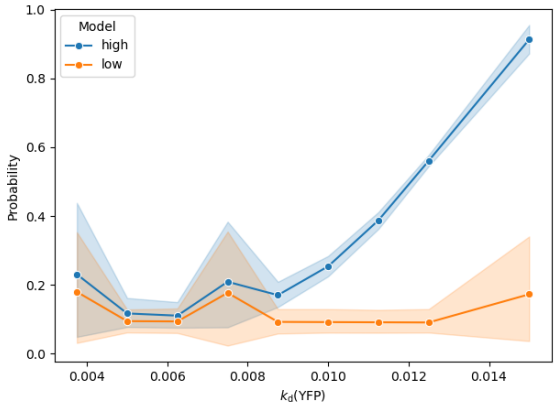


Figure 4: Average Probability of Failure vs kd of YFP

These findings illustrate a critical trade-off: degradation rates that optimize robustness in one state inherently reduce stability in the other. This emphasizes the need for a balanced degradation profile across YFP, LacI, and TetR to minimize the overall failure probability across both operational states. Thus, we use a set of metrics to assign "scores" to each parameter set. The goal of these metrics is to penalize both high probabilities and imbalanced probabilities. Our metrics include a simple sum ($p_{\text{high}} + p_{\text{low}}$), average plus difference ($(p_{\text{high}} + p_{\text{low}})/2 + \text{abs}(p_{\text{high}} - p_{\text{low}})$), sum of squares ($p_{\text{high}}^2 + p_{\text{low}}^2$) and cubes ($p_{\text{high}}^3 + p_{\text{low}}^3$), and several others. An automated analysis found that for 20 metrics over all parameter combinations, six combinations for YFP, LacI, TetR outperformed the rest:

- (0.00375, 0.00375, 0.00375) was selected by 5 metrics
- (0.00375, 0.015, 0.00375) was selected by 5 metrics
- (0.015, 0.00375, 0.015) was selected by 5 metrics
- (0.00625, 0.00625, 0.005) was selected by 2 metrics
- (0.015, 0.00375, 0.00375) was selected by 2 metrics
- (0.015, 0.015, 0.00375) was selected by 1 metric

The combination (0.00375, 0.00375, 0.00375), selected by five metrics, was manually evaluated to be the best overall combination, as it provides a reasonable balance between probability of failure for both the high and low models.

DISCUSSION AND FUTURE WORK

Currently, our method relies on exhaustive enumeration of degradation rate combinations, which scales poorly with increasing model complexity or additional tunable parameters. To address this limitation, we aim to incorporate parameter synthesis techniques, which can symbolically derive sets of parameters that satisfy specified probabilistic constraints. This approach would allow us to efficiently explore the parameter space and identify robust configurations without the computational cost of brute-force analysis.

Furthermore, we plan to extend this methodology to more complex genetic circuits, where interactions among additional components further complicate system dynamics. By integrating probabilistic model checking with parameter synthesis and experimental feasibility constraints, we envision a computational pipeline capable of guiding the design of large-scale synthetic systems with tunable robustness properties.

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation under Grant Nos. 2422206 and 1856733. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Landon Taylor also acknowledges partial support from the University of Utah CMCR Seed Grant.

REFERENCES

[1] Gardner, T. S., Cantor, C. R., and Collins, J. J. Construction of a genetic toggle switch in Escherichia coli. *Nature 403*, 6767 (Jan. 2000), 339–342.

[2] Kwiatkowska, M., Norman, G., and Parker, D. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Computer Aided Verification* (Berlin, Heidelberg, 2011), G. Gopalakrishnan and S. Qadeer, Eds., Springer Berlin Heidelberg, pp. 585–591.

[3] 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 8*, 7 (July 2019), 1560–1563.

Stochastic analysis of Single-Cell Dynamics and Population-Level Behavior in Biosensors

Bingqing Hu[1], **Joshua Jeppson**[1], **Micha Claydon**[2], **Thomas E. Gorochowski**[2], **Zhen Zhang**[1], **Lukas Buecherl**[3]
[1]Electrical and Computer Engineering, Utah State University, USA; [2] School of Biological Sciences, University of Bristol, UK; [3] Biological Engineering, Utah State University, USA
lukas.buecherl@usu.edu

INTRODUCTION

High-performance biosensors require rapid and reliable activation, especially when detecting transient or low-abundance signals. Specifically, the duration for a biosensor to reach a functional output level determines how quickly downstream processes are triggered and constrains the practicality of its use. In many biosensing systems, stochastic molecular events play a key role in affecting the variability seen for activation, with some cells reaching target levels much earlier than others [2]. Understanding these rare, early activations could be key for improving performance.

In this work, we use an experimentally characterized repression-based genetic circuit based on CRISPRi [1] as a case study to investigate how quickly an output can accumulate when considering thestochastic nature of the biochemical reactions involved. Although this system is designed to remain inactive under repression, our analysis focuses on cases where it *fails* to do so, i.e, when stochastic fluctuations push the output above a functional threshold. To compare the timing of activation on trajectories, we define a nominal benchmark for activation at a 10 nM output threshold, chosen because this concentration is sufficient to trigger further signal amplification methods. Using this reference point, we examine how quickly individual stochastic runs cross this threshold and quantify the probability of early activation events. Our analysis addresses two questions: **(1)** at the single-cell level, which molecular reactions most influence early threshold crossing, and **(2)** at the population level, how large must a population be for at least one cell to reach the threshold within a given time window.

By combining stochastic simulations, reaction frequency analysis, and probabilistic model checking, we are able to link the dynamics of molecular events to predictions of activation timing, providing a framework for evaluating and optimizing biosensor responsiveness.

METHODS

The CRISPRi circuit we studied consists of four transcriptional units (Fig. 1). The first (Fig. 1, lower left) is regulated by aTc and produces the dCas9 protein (orange). Throughout our analysis, aTc concentration was kept constant to ensure continuous expression of dCas9. The second unit (Fig. 1, upper left) is regulated by IPTG and produces a guide RNA (pink) that binds to dCas9 and represses the third transcriptional unit. When the third unit is not repressed, it produces another guide RNA that forms a complex with dCas9 to activate a fourth unit, leading to the expression of the output Y_C. The third and fourth units (Fig. 1, lower right) are represented as an engineered gene (blue). High IPTG concentrations maintain low Y_C production, whereas low IPTG levels allow activation.

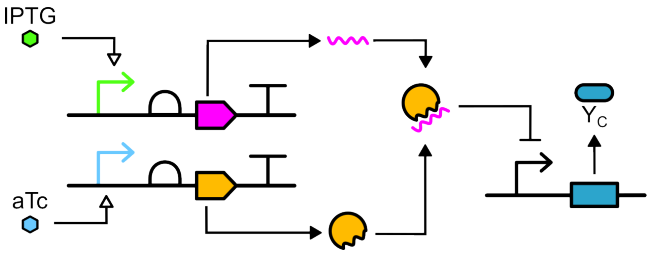


Figure 1: Schematic of the repression-based CRISPRi circuit.

We modeled the circuit in its repressed state (IPTG = 1000), where the biosensor should remain inactive. This setup allows us to address two complementary questions: **(1)** the timing and sequence of reactions that lead to premature threshold crossings at the single-cell level, and **(2)** how quickly can the circuit reach a functional output level 10 nM at the population level, a metric that parallels the desired rapid response performance in an active sensing scenario.

The reaction network was encoded as a *chemical reaction network* (CRN) and implemented in Julia for deterministic ODE simulations [1], in PRISM [4] as a *continuous-time Markov chain* (CTMC) for both stochastic simulations and probabilistic model checking in Wayfarer [3]. Wayfarer is a tool that, given a target region, overlays all possible system states into a vector space and invokes the notion of distance within that vector space to prioritize pathways more likely to reach a target. Additionally, Wayfarer is capable of creating nested subspaces in the model's vector space, representing sequences of raction firings in a CRN, which are *iteratively* prioritized, allowing it to handle complex CRNs such as the CRISPRi model analyzed in this work.

We simulated the repressed state in PRISM with 10 stochastic runs, with Fig.2 showing how the output Y_C concentration varies over time. Due to stochastic variation, some runs reached the threshold quicker (e.g., Run 3) than others (e.g., Run 4). To pinpoint the molecular origins of this variability, we analyzed reaction firing frequencies in the short window before threshold crossing. This identified specific reaction sequences that either accelerated or delayed activation, revealing kinetic bottlenecks and dominant activation paths at the single-cell level.

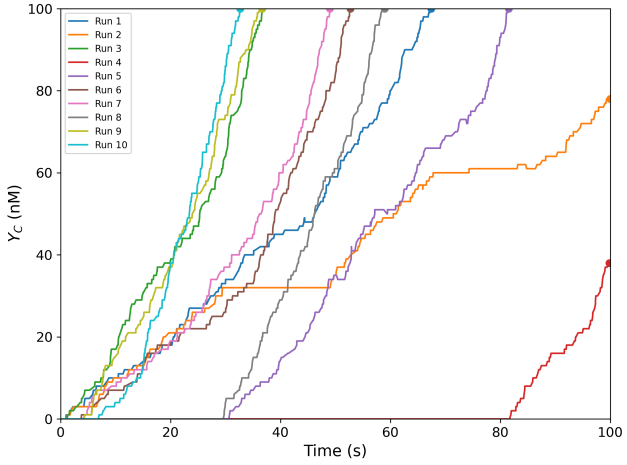


Figure 2: Stochastic Y_C trajectories in the repressed state (IPTG = 1000)

For our second analysis, we use the Wayfarer probabilistic model checking tool to compute the minimum probability ($P_{\min}$) of reaching 10 nM within a defined time limit of 10 seconds. This yields a population-level interpretation by estimating how many cells are needed for at least one to cross the threshold within the time window. Combined with single-cell reaction analysis, this provides a complementary framework for linking molecular events to population-scale outcomes.

RESULTS

The stochastic runs showed that threshold crossing was primarily driven by early firing of the transcription reaction that produced Y_C's mRNA, followed shortly by the translation of mRNA into Y_C, which rapidly increased output levels. Runs lacking early transcription activity either reached the 10 nM threshold much later or failed to do so within the observation window, indicating that mRNA transcription initiation is the main kinetic bottleneck for premature activation.

Moving to the population-level analysis, Wayfarer probabilistic model checking computed the minimum probability ($P_{\min}$) of reaching the 10 nM threshold within 10 seconds under repression. The result $P_{\min} = 5.29 \times 10^{-6}$, indicates that a population of roughly 5.3 million cells would, on average, contain a single cell that crosses the threshold in this short time frame. For reference, there are ~1 billion cells in a saturated 1 mL culture). Increasing the threshold to 20 or 30 nM reduced $P_{\min}$ by several orders of magnitude (Table1), underscoring the rarity of such rapid activations.

Table 1: Minimum probability ($P_{\min}$) of reaching a given Y_C threshold within time T, computed by *Wayfarer*.

Y_C (nM)	T (s)	$P_{\min}$
30	10	3.37×10^{-14}
20	10	1.90×10^{-10}
10	10	5.29×10^{-6}

DISCUSSION AND FUTURE WORK

We have combined single-cell stochastic simulation with population-level probabilistic analysis to determine both the specific molecular events that trigger rapid activation and the likelihood that these events occur within a given time frame. At the single-cell level, stochastic simulations and reaction frequency analysis showed that early threshold crossing is primarily driven by rapid initiation of Y_C mRNA transcription, immediately followed by translation into Y_C. This sequence represents a clear kinetic bottleneck: When it occurs early, output levels rise rapidly; when it is delayed, activation is slow or absent within the observation window.

At the population level, probabilistic model checking provided a quantitative link between these rare molecular trajectories and their likelihood of appearing in real cell populations. The analysis showed that even under repression, a sufficiently large population could contain a cell that reached the amplification threshold 10 nM in seconds.

Beyond this case study, the pipeline offers a generalizable approach to evaluate both failure risks and activation speeds in biosensors. Future applications could invert the problem: rather than preventing premature activation, circuit designs could intentionally exploit fast-responding rare trajectories to achieve ultra-rapid sensing. Integrating these computational insights with single-cell measurements will be a key to validating and extending this methodology to a broader class of synthetic circuits.

ACKNOWLEDGMENTS

B.H. was supported from the Utah State University Integrated Team Research Seed Grant. T.E.G. was supported by a Royal Society University Research Fellowship grant URF\R\221008. J.J was supported by the National Science Foundation under Grant Nos. 2422206 and 1856733. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Claydon, M. Y. *Exploring the design of biological circuits using dCas-based regulators.* PhD thesis, University of Bristol, 2025.

[2] Ham, L., Coomer, M. A., Öcal, K., Grima, R., and Stumpf, M. P. A stochastic vs deterministic perspective on the timing of cellular events. *Nature Communications 15*, 1 (2024), 5286.

[3] Jeppson, J., Taylor, L., Hu, B., and Zhang, Z. Reasoning about rare-event reachability in stochastic vector addition systems via affine vector spaces, 2025.

[4] Kwiatkowska, M., Norman, G., and Parker, D. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Computer Aided Verification* (Berlin, Heidelberg, 2011), G. Gopalakrishnan and S. Qadeer, Eds., Springer Berlin Heidelberg, pp. 585–591.

SYNBICT 2.0: Scalable Synthetic Biology Part Annotation at Speed

Chunxiao Liao
University of Colorado Boulder
Boulder, CO, USA
Chunxiao.Liao@colorado.edu

Chris Myers
University of Colorado Boulder
Boulder, CO, USA
Chris.Myers@colorado.edu

1 INTRODUCTION

Accurate annotation of synthetic biology parts from engineered plasmids is essential for downstream tasks such as circuit construction and simulation. SYNBICT 1.0 [8] provides a pipeline for part annotation and circuit inference. However, SYNBICT 1.0 suffers from slow processing speed and inefficient memory usage. This work presents a two-fold improvement over SYNBICT 1.0. First, it enhances scalability while maintaining the same level of accuracy and sensitivity. Second, it introduces a similarity search feature that enables the detection of similar parts within plasmids, which can assist in determining the function of sequences that do not exactly match any parts in the library. We evaluated six related tools and compared their performance to SYNBICT 1.0 in terms of runtime, memory consumption, and annotation accuracy. The proposed solution achieves up to 100× speed improvement and 100× reduction in memory usage, without compromising accuracy.

What defines a part annotation? In a narrow sense, it refers to an exact match, where the part is a complete substring of the plasmid with 100% identity and no single-nucleotide polymorphisms (SNPs) in the aligned region. This is the definition currently used by SYNBICT 1.0. However, due to the complexity of real-world biological systems [6], parts are not always perfect matches. Some parts may have been intentionally modified to adapt performance or improve synthesis yield (ex. codon optimization). To address these challenges, this work accounts for both scenarios. In addition to exact matching, it introduces a similarity search that detects similar parts, while maintaining the same runtime and memory efficiency.

2 SYNBICT 2.0

Benchmark Data

The benchmark dataset consists of 50,000 plasmids downloaded from the Addgene repository [1] and converted to the SBOL format, with each initially annotated using SnapGene https://www.snapgene.com/. These plasmids were also previously annotated by SYNBICT 1.0 [5]. For benchmarking, each SBOL file was processed using SYNBICT 1.0, and the resulting annotated SBOL files served as the benchmark reference.

Memory Profile of SYNBICT 1.0

The goal of this step is to identify bottlenecks in SYNBICT 1.0. To do this, we encapsulated the annotation function, used consistent parameters, and applied a Python memory profiler to analyze memory usage line by line. We identified two major bottlenecks. First, memory usage increases linearly with the number of features in the SBOL document, as shown in Figure 1. This is due to the storage of the document as a list. When processing 5,000 features, the list consumes approximately 700 MB of memory. For larger feature libraries, this becomes unsustainable. Moreover, most SBOL documents contain redundant information, and for the purpose of annotation, neither full sequences nor extensive metadata are necessary. These can be eliminated or reduced to a compact dictionary representation. The corresponding memory profile is shown in Figure 2.

The second bottleneck lies in the mapping step, which is both slow and memory-intensive. Mapping a single plasmid to the feature library takes approximately 15 seconds and consumes around 800 MB of memory. The memory profiling results of SynBICT 1.0 are shown in Figure 2, with the high-memory regions highlighted.

End-End Testing Suite

To ensure a fair comparison, we implemented the same input-output logic as SYNBICT 1.0, taking SBOL as input and producing SBOL as output, while replacing only the annotation function with six alternative algorithms, BWA [3], Minimap2 [4], Bowtie2 [2], BLASTN [12], VSEARCH [9], and MMseq2 [10]. The revised workflow is illustrated in Figure 4. It begins by converting the feature library into a FASTA file and extracting only the necessary metadata into a dictionary for annotation. The FASTA file is then indexed, followed by alignment using different tools to generate Sequence Alignment/Map (SAM) files. These SAM files are subsequently converted back into SBOL format. In total, seven methods (including SYNBICT 1.0) were used to process the Addgene plasmid dataset. We then compared their output, runtime, and memory usage.

Results Analysis

The core annotation result consists of the feature ID, start position, end position, and strand. The new method outputs results in SAM format, which is more informative and can be easily converted to Binary Alignment Map (BAM) file for visualization in the *Integrative Genomics Viewer* (IGV) [11]. Figure 3 presents the Minimap2 memory profile, which can be compared with Figure 2. Table 1 compares mapping tools under an exact-match parameter setting, reporting mean runtime (s), mean memory usage (MB), the mean±SD number of annotations detected, and intermediate library file size. Table 2 presents the corresponding comparison under similar parameter settings. Note that SynBict 1.0 does not produce an intermediate file. Across tools (BWA, BLASTN, Minimap2), exact-match annotation yields are essentially identical; the small residual variance likely reflects parameter choices or downstream filtering. BWA is the fastest, whereas BLASTN uses the least memory and produces the smallest intermediate files. BLASTN and Minimap2 are more sensitive to short-fragment mappings, while BWA tends to miss very short segments under current parameter setting, this warrants further testing. Overall, because BLASTN leads in three dimensions (memory, file size, short-fragment sensitivity), we provisionally select it as the best tool for exact matching. Analysis under 'similar' (non-exact) settings is deferred to future work. Figure 5 shows an IGV visualization of similar parts mapped to the plasmid using BWA, with each track corresponding to a different parameter setting. For similarity-based mapping, we will integrate biological knowledge (e.g., start codons and conserved regions), protein-level alignments, and sequence analysis models (e.g., HMMER [7]) to develop a multi-criteria filtering strategy that goes beyond a simple identity-threshold rule.

3 CONCLUSION

We enhanced the SYNBICT tool by significantly reducing memory usage and improving processing speed, without compromising annotation performance. Additionally, we introduced a similarity search function, enabling the tool to curate larger feature libraries and incorporate data from external databases and repositories. These improvements also support more robust downstream analyzes, such as predicted part filtering and circuit profiling. The updated SYNBICT implementation, along with the corresponding test suite and benchmark results, is available on GitHub: https://github.com/SD2E/SYNBICT/tree/SYNBICT2.

4 ACKNOWLEDGMENT

This work was supported by the National Science Foundation under Grant No. 2231864. 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] Kamens, J. The addgene repository: an international nonprofit plasmid and data resource. *Nucleic acids research 43*, D1 (2015), D1152–D1157.
[2] Langmead, B., and Salzberg, S. L. Fast gapped-read alignment with bowtie 2. *Nature methods 9*, 4 (2012), 357–359.
[3] Li, H. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997* (2013).
[4] Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics 34*, 18 (2018), 3094–3100.
[5] Mante, J. *Promotion of Data Reuse in Synthetic Biology*. University of Colorado at Boulder, 2022.
[6] McGuffie, M. J., and Barrick, J. E. plannotate: engineered plasmid annotation. *Nucleic acids research 49*, W1 (2021), W516–W522.
[7] Potter, S. C., Luciani, A., Eddy, S. R., Park, Y., Lopez, R., and Finn, R. D. Hmmer web server: 2018 update. *Nucleic acids research 46*, W1 (2018), W200–W204.
[8] Roehner, N., Mante, J., Myers, C. J., and Beal, J. Synthetic biology curation tools (synbict). *ACS synthetic biology 10*, 11 (2021), 3200–3204.
[9] Rognes, T., Flouri, T., Nichols, B., Quince, C., and Mahé, F. Vsearch: a versatile open source tool for metagenomics. *PeerJ 4* (2016), e2584.
[10] Steinegger, M., and Söding, J. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology 35*, 11 (2017), 1026–1028.
[11] Thorvaldsdóttir, H., Robinson, J. T., and Mesirov, J. P. Integrative genomics viewer (igv): high-performance genomics data visualization and exploration. *Briefings in bioinformatics 14*, 2 (2013), 178–192.
[12] Ye, J., McGinnis, S., and Madden, T. L. Blast: improvements for better sequence analysis. *Nucleic acids research 34*, suppl_2 (2006), W6–W9.

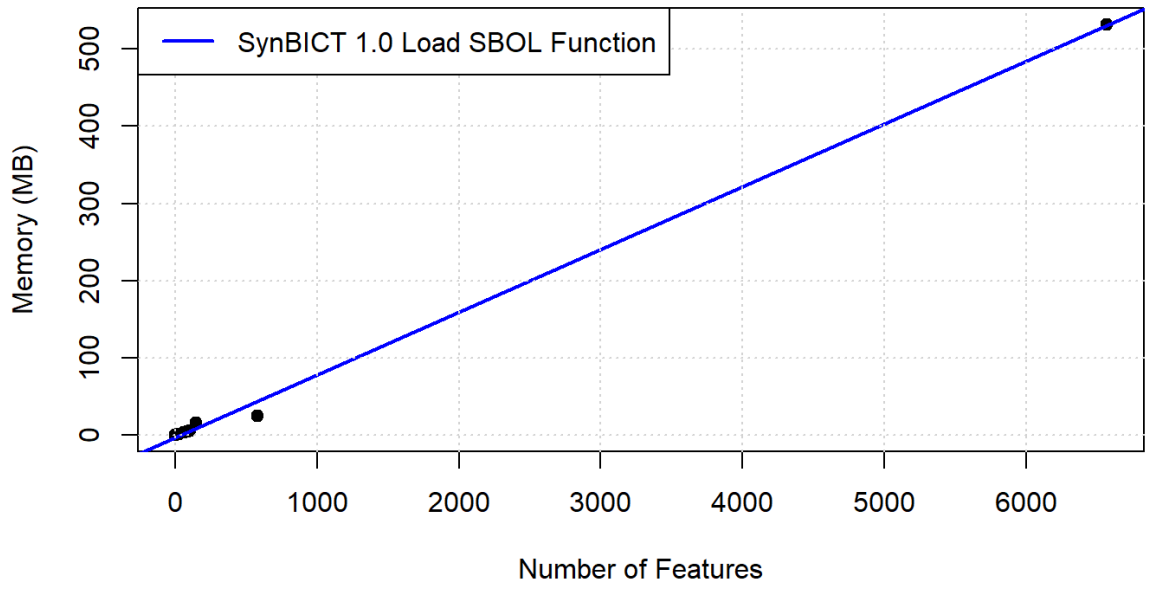


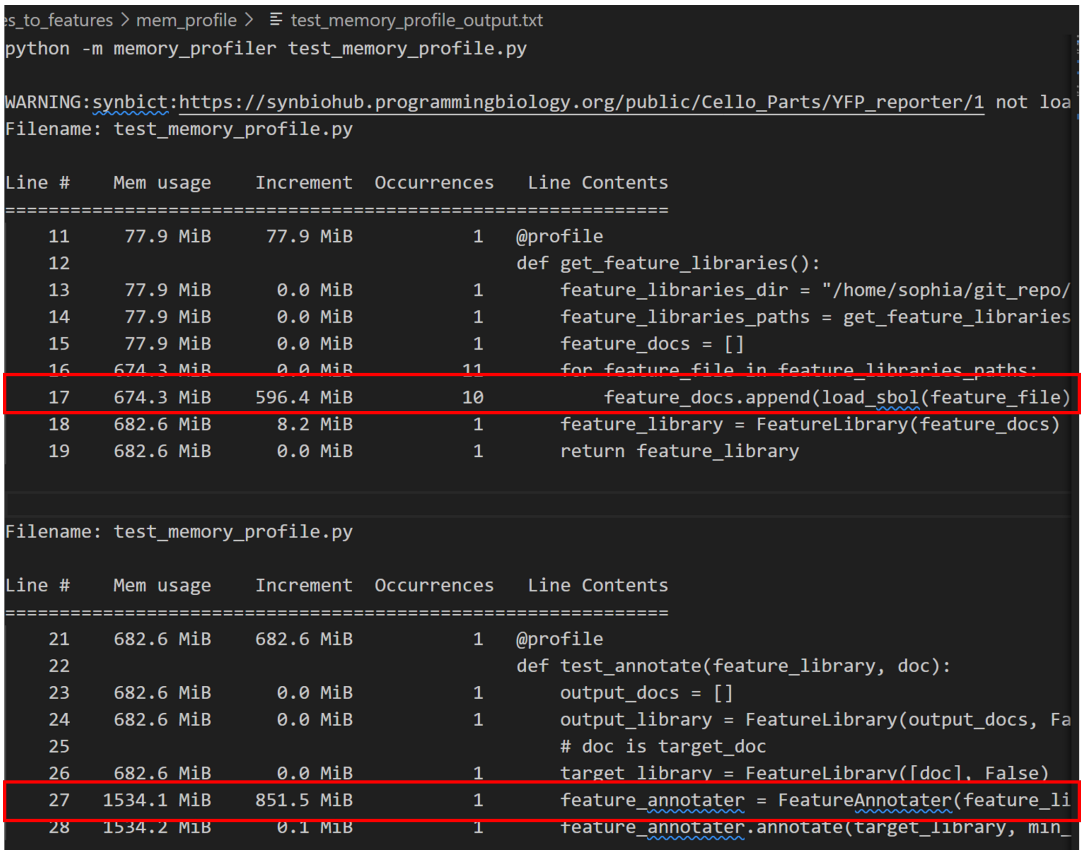
Figure 1: Memory usage increases linearly with the number of features.

Table 1: Performance Comparison of Annotation Methods Under Exact-match Parameter Setting

Tool	Time (s)	Memory (MB)	# Annotations	File Size (MB)
SYNBICT 1.0 exact	15.2±2.5	810.58±.06	1.43±1.48	n/a
Minimap2 [4] exact	.107±.03	5.9±.04	1.35±1.41	11.02
BWA [3] exact	**.076±.09**	15.01±.15	1.44±1.51	7.34
BLASTN [12] exact	.21±.09	**3.37±.09**	1.47±1.53	**3.87**
Bowtie2 [2] exact	.26±.16	4.17±0	.21±.7	14.6
VSEARCH [9] exact	10.55±8.70	3.07±.01	.59±.8	14.6

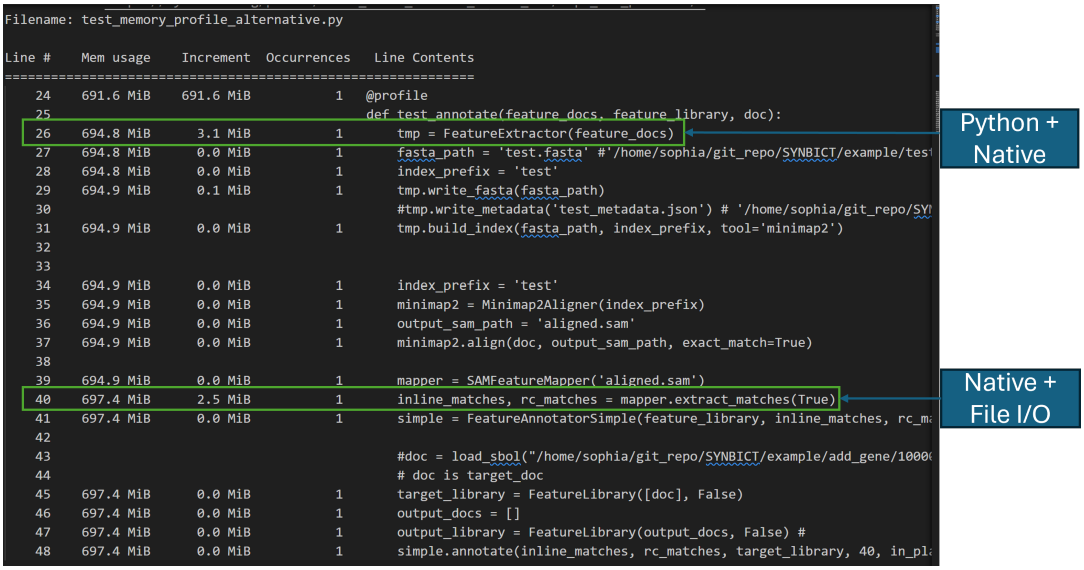
Table 2: Performance Comparison of Annotation Methods Under Similar-match Parameter Settings

Tool	Time (s)	Memory (MB)	# Annotations	File Size (MB)
SYNBICT 1.0 similar	13.02±1.35	811.87±.61	2.01±1.77	n/a
Minimap2 similar	.21±.11	6.01±.09	7.13±3.72	11.02
BWA similar	**.15±10**	15.04±.13	8.2±5.2	7.34
BLASTN similar	.28±.13	**3.41±.14**	10.19±5.79	**3.87**
Bowtie2 similar	.42±.26	4.39±.15	9.99±5.91	14.6
VSEARCH similar	5.14±5.08	3.07±0	1.19±.61	14.6
MMseqs2 [10]	9.57±2.31	3.2±.05	2.76±2.08	5.7



```
s_to_features > mem_profile > test_memory_profile_output.txt
python -m memory_profiler test_memory_profile.py

WARNING:synbict:https://synbiohub.programmingbiology.org/public/Cello_Parts/YFP_reporter/1 not loa
Filename: test_memory_profile.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    11     77.9 MiB     77.9 MiB           1   @profile
    12                                         def get_feature_libraries():
    13     77.9 MiB      0.0 MiB           1       feature_libraries_dir = "/home/sophia/git_repo/
    14     77.9 MiB      0.0 MiB           1       feature_libraries_paths = get_feature_libraries
    15     77.9 MiB      0.0 MiB           1       feature_docs = []
    16    674.3 MiB      0.0 MiB          11       for feature_file in feature_libraries_paths:
    17    674.3 MiB    596.4 MiB          10           feature_docs.append(load_sbol(feature_file)
    18    682.6 MiB      8.2 MiB           1       feature_library = FeatureLibrary(feature_docs)
    19    682.6 MiB      0.0 MiB           1       return feature_library


Filename: test_memory_profile.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    21    682.6 MiB    682.6 MiB           1   @profile
    22                                         def test_annotate(feature_library, doc):
    23    682.6 MiB      0.0 MiB           1       output_docs = []
    24    682.6 MiB      0.0 MiB           1       output_library = FeatureLibrary(output_docs, Fa
    25                                             # doc is target_doc
    26    682.6 MiB      0.0 MiB           1       target_library = FeatureLibrary([doc], False)
    27   1534.1 MiB    851.5 MiB           1       feature_annotater = FeatureAnnotater(feature_li
    28   1534.2 MiB      0.1 MiB           1       feature_annotater.annotate(target_library, min_
```

Figure 2: Memory profile of SYNBICT 1.0.



```
Filename: test_memory_profile_alternative.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    24    691.6 MiB    691.6 MiB           1   @profile
    25                                         def test_annotate(feature_docs, feature_library, doc):
    26    694.8 MiB      3.1 MiB           1       tmp = FeatureExtractor(feature_docs)
    27    694.8 MiB      0.0 MiB           1       fasta_path = 'test.fasta' #'/home/sophia/git_repo/SYNBICT/example/tes
    28    694.8 MiB      0.0 MiB           1       index_prefix = 'test'
    29    694.9 MiB      0.1 MiB           1       tmp.write_fasta(fasta_path)
    30                                             #tmp.write_metadata('test_metadata.json') # '/home/sophia/git_repo/SY
    31    694.9 MiB      0.0 MiB           1       tmp.build_index(fasta_path, index_prefix, tool='minimap2')
    32
    33
    34    694.9 MiB      0.0 MiB           1       index_prefix = 'test'
    35    694.9 MiB      0.0 MiB           1       minimap2 = Minimap2Aligner(index_prefix)
    36    694.9 MiB      0.0 MiB           1       output_sam_path = 'aligned.sam'
    37    694.9 MiB      0.0 MiB           1       minimap2.align(doc, output_sam_path, exact_match=True)
    38
    39    694.9 MiB      0.0 MiB           1       mapper = SAMFeatureMapper('aligned.sam')
    40    697.4 MiB      2.5 MiB           1       inline_matches, rc_matches = mapper.extract_matches(True)
    41    697.4 MiB      0.0 MiB           1       simple = FeatureAnnotatorSimple(feature_library, inline_matches, rc_m
    42
    43                                             #doc = load_sbol("/home/sophia/git_repo/SYNBICT/example/add_gene/1000
    44                                             # doc is target_doc
    45    697.4 MiB      0.0 MiB           1       target_library = FeatureLibrary([doc], False)
    46    697.4 MiB      0.0 MiB           1       output_docs = []
    47    697.4 MiB      0.0 MiB           1       output_library = FeatureLibrary(output_docs, False) #
    48    697.4 MiB      0.0 MiB           1       simple.annotate(inline_matches, rc_matches, target_library, 40, in_pl
```


Figure 3: Memory profile of SYNBICT 2.0 with minimap2.

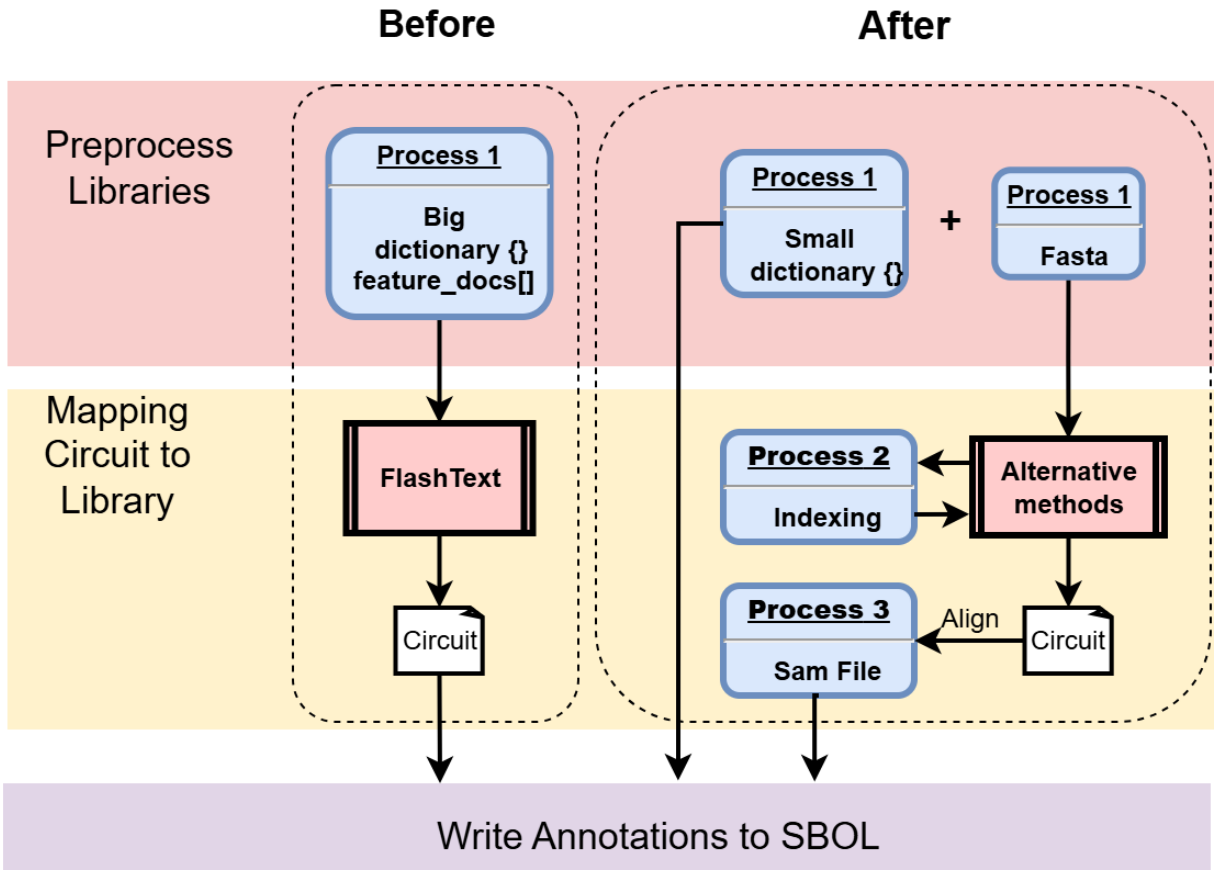


Figure 4: Schematic figure of the Overall structure before and after.

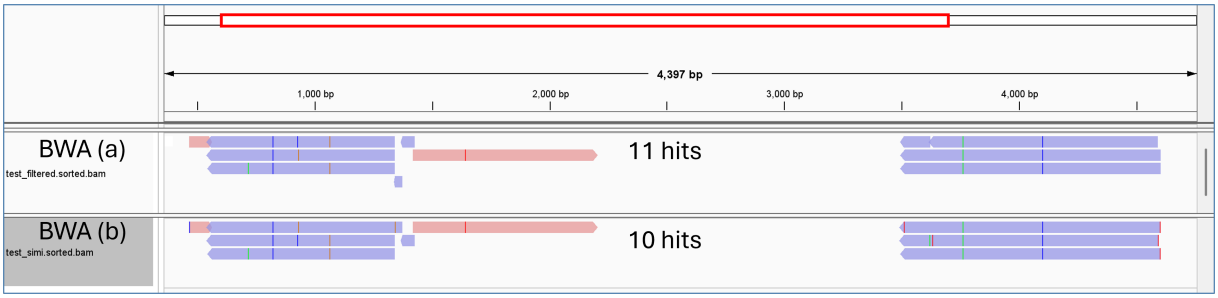


Figure 5: IGV visualization of two groups of alignment hits generated using two different BWA parameter settings on plasmid Addgene 127789.

LaNVis: Biological Constraint-based Large Network Visualizer

Yangruirui Zhou
Boston University
Boston, MA, USA
yrrzhou@bu.edu

Christopher A. Voigt
Massachusetts Institute of Technology
Cambridge, MA, USA
cavoigt@gmail.com

Douglas Densmore
Boston University
Boston, MA, USA
dougd@bu.edu

1 INTRODUCTION

The design and implementation of synthetic genetic circuits have advanced over the past decade, driven by the emergence of genetic compilers and standard biological parts [1]. However, translating complex digital logic into reliable biological implementations remains a challenge due to the inherent complexity of mapping digital logic onto biomolecules. Tools such as Cello [2] enable users to describe digital logic using Verilog and automatically translate it into DNA sequences. While these approaches allow individual cells to be programmed, their capacity is inherently limited: freely diffusing molecules can cause signal interference, and the metabolic burden of expressing regulatory components can impair cell stability. To scale up, computation must be distributed across multiple cells, introducing new challenges in circuit partitioning, submodule design, and particularly in visualization.

Several prior works have addressed parts of this challenge. Oriole [3] provides a constraint-based graph partitioning framework for distributing large-scale logic circuits into modules under biological constraints, but it does not produce HDL code or module-level schematics. Al-Radhawi et al.'s Distributed Boolean Computation (DBC) framework [4] can automatically generate multicellular Boolean logic designs from a truth table combined with biological constraints, and visualizes the resulting design as cell-level schematics. While these methods address "how to partition" or "how to design" multicellular circuits from scratch, they do not focus on interpreting or restructuring an existing HDL description.

In this work, we present LaNVis, a tool that bridges the gap between hardware-level circuit descriptions and biologically constrained multicellular implementations. LaNVis combines constraint-based partitioning with scalable visualization and module reuse, producing hierarchical schematics that clarify circuit structure at multiple abstraction levels. By integrating seamlessly with downstream platforms such as Cello, LaNVis provides a unified workflow from HDL descriptions to biologically realizable genetic circuits.

2 METHOD

LaNVis takes as input a logic circuit described in Verilog. While the original circuit can be specified using any logic operators, after partitioning, the subcircuits assigned to individual cells are restricted to NOR2 and NOT gates to match biologically realizable primitives.

Partitioning is performed using a constraint-aware algorithm derived from Oriole. Users can specify constraint values such as the maximum number of gates per cell, the number of intercellular communication channels, and the type of in/out degree counting (as defined in Oriole). The choice of constraint values has a decisive influence on the modular architecture, directly shaping both biological feasibility and circuit readability.

After partitioning, LaNVis identifies and merges functionally equivalent cells. Two cells are considered equivalent if they exhibit the same internal logic structure, contain the same number of gates, and use the same set of communication channels. Importantly, they must also share identical input and output molecular species, since distinct molecules are required to implement different communication signals in practice. This merging step enables biological reuse by allowing one genetic construct to be replicated across multiple cell populations, thus reducing the need for redundant genetic editing and lowering implementation costs.

The resulting design is visualized hierarchically. At the top level, LaNVis generates a schematic showing the connectivity among reusable cell modules, with each module represented as a black box. For each module, gate-level schematics and corresponding Verilog files are produced using existing tools such as Yosys (for Verilog synthesis and JSON generation) and Netlistsvg (for schematic rendering). The complete multicellular system is exported in Verilog format, ensuring compatibility with downstream genetic design automation tools such as Cello.

3 DISCUSSION

Compared to prior frameworks such as Oriole, which focuses on partitioning without schematic output, and Distributed Boolean Computation (DBC), which generates multicellular designs from truth tables, LaNVis emphasizes the interpretation and restructuring of pre-existing designs while maintaining compatibility with biological constraints. By producing hierarchical, layout-aware schematics directly from

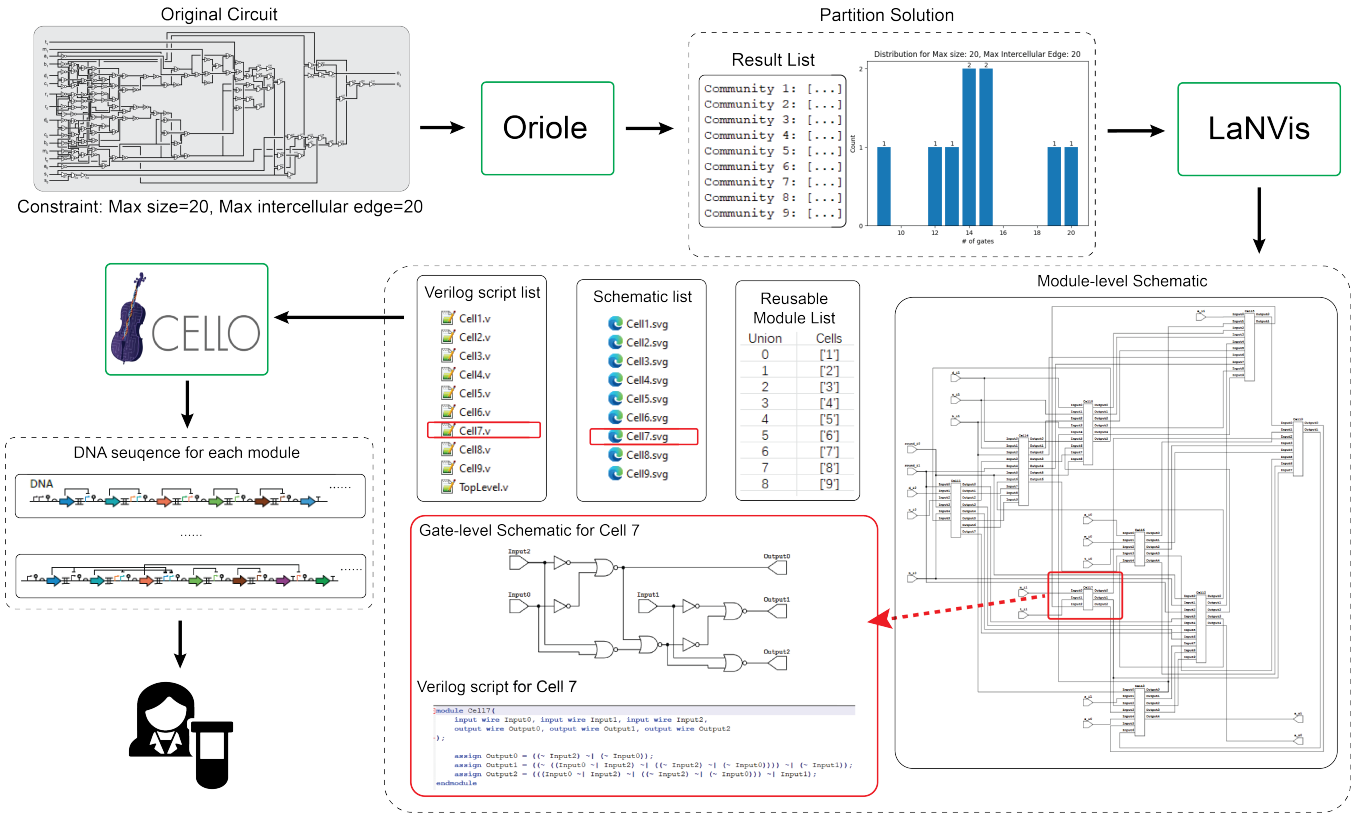


Figure 1: End-to-end workflow for compiling a digital logic circuit into DNA sequences. The input Verilog file and bio-related constraints are first processed by Oriole to generate a feasible circuit partitioning solution. The results are visualized with a distribution plot indicating the number of gates allocated to each cell. LaNVis then takes the partition result and produces module-level and gate-level schematics, a Verilog file list, and a reusable cell library. These outputs are compatible with Cello, which translates the circuit into a DNA sequence list for experimental implementation. This workflow highlights the seamless integration of LaNVis with existing tools and its contribution to biologically realizable multicellular designs.

Verilog representations, including large-scale circuits, LaNVis enables researchers to visualize, understand, and debug multicellular systems without reconstructing modules.

The intuitive visualizations produced by LaNVis improve circuit readability, reduce wiring congestion, and clarify module boundaries, enabling both computational and experimental researchers to efficiently assess circuit logic. This accessibility lowers technical barriers for experimental biologists, allowing them to independently explore complex designs, iterate more rapidly, and communicate circuit behavior across disciplinary boundaries. As a result, LaNVis supports more informed design decisions, facilitates troubleshooting, and enhances integration with downstream genetic design automation tools such as Cello.

In the long term, this approach could accelerate the development of real-world applications where complex logic circuits are critical. By providing a clear, structured, and biologically aware representation of large-scale genetic circuits, LaNVis establishes a foundational infrastructure for scalable and interpretable synthetic biology design.

REFERENCES

[1] Jai P Padmakumar, Jessica J Sun, William Cho, Yangruirui Zhou, Christopher Krenz, Woo Zhong Han, Douglas Densmore, Eduardo D Sontag, and Christopher A Voigt. Partitioning of a 2-bit hash function across 66 communicating cells. *Nature Chemical Biology*, 21(2):268–279, 2025.

[2] Alec AK Nielsen, Bryan S Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A Strychalski, David Ross, Douglas Densmore, and Christopher A Voigt. Genetic circuit design automation. *Science*, 352(6281):aac7341, 2016.

[3] Yangruirui Zhou, Christopher A Voigt, and Douglas Densmore. Constraint-based sub-graph partitioning for multi-cellular biological networks. *IEEE Transactions on Computational Biology and Bioinformatics*, 2025.

[4] M Ali Al-Radhawi, Anh Phong Tran, Elizabeth A Ernst, Tianchi Chen, Christopher A Voigt, and Eduardo D Sontag. Distributed implementation of boolean functions by transcriptional synthetic circuits. *ACS Synthetic Biology*, 9(8):2172–2187, 2020.

GeneForge: Agentic AI Server for Design Automation

Jordan Graves
Harvard Extension School
Boston, USA
jordanlgraves@icloud.com

Chunxiao Liao
University of Colorado Boulder
Boulder, USA
Chunxiao.Liao@colorado.edu

Chris Myers
University of Colorado Boulder
Boulder, USA
Chris.Myers@colorado.edu

1 INTRODUCTION

GeneForge is an AI-driven platform for biodesign automation (BDA) that utilizes autonomous agents to streamline synthetic biology workflows, from initial design specification through modeling and simulation. Motivated by the labor-intensive nature of genetic circuit engineering, the coordination burden of numerous specialized tools, and the growing interest in scientific discovery workflows powered by AI agents, GeneForge leverages a large language model (LLM) as a central reasoning engine. This agent plans and executes tasks, designs genetic circuits, generates novel biological parts, and conducts kinetic simulations using integrated computational biology tools with minimal human oversight.

2 GENEFORGE AGENTS

The platform integrates diverse bio-design resources, including the Cello logic circuit suite [4], ProD promoter engineering [7], Tellurium biochemical modeling [1], BioNumbers database [3], and SynBioHub genetic part repository [2], etc. These tools are unified under a single tool specification compatible with modern LLMs.

A web interface allows a user to provide natural language requests which are routed to the LLM with a managed session state. After generating and interpreting the intent of the user through "thinking" tokens, the LLM may specify function calls and associated parameters defined within the tool specifications. The system executes the tool call with the specified parameters, potentially modifying the session state. The result of the tool call (e.g., a list of parts or context about E. coli) along with any changes to the session state is passed back to the LLM Client, which now has richer context to continue reasoning and responding. This loop allows the AI agent to chain tool calls, maintain memory of actions, and achieve more complex design goals. Figure 1 shows an example workflow for part selection. After that, the AI agent proceeds to select the next tool, and continues through each step until the simulation is completed.

GeneForge learns from two automated signals. First, the rollback-and-retry events create self-supervised preference data for Direct Preference Optimization (DPO) [5]: when a tool call fails, the agent rewinds to the last stable state and explores alternative tools/parameters. Upon successful correction, a trace is logged, yielding (preferred, rejected) pairs that train the policy (model weights) to improve tool calling. Second, Group-Relative Policy Optimization (GRPO) [6] drives end-to-end performance using programmatic rewards with no binary labels: starting from natural-language prompts that specify a workflow or target outcome, the system samples multiple high-temperature rollouts per prompt, capturing full session logs (tool calls, parameters, outputs, decisions). Each rollout receives a scalar reward computed from the message stream which includes model choices, tool outputs and session state history. Within each prompt's group these rewards are normalized to advantages to update the policy with KL-regularized GRPO. In short, DPO teaches preference structure from real failure recoveries, while GRPO optimizes long-horizon behavior from verifiable, low-variance reward signals. Together, these methods yield a policy that both avoids prior failure modes and improves performance on task-level objectives. A fine-tuning workflow is shown in Figure 2.

3 CONCLUSION

We present GeneForge, an open-source AI agent prototype that automates the core stages of genetic circuit design, including part selection, circuit construction, and behavior simulation, and combines this with verifiable outcomes to drive continuous model improvement through integrated preference-based and reinforcement learning loops. Designed for adaptability, GeneForge can be extended and tested across a range of synthetic biology applications, offering a scalable foundation for future AI-driven design in synthetic biology. The source code is available on GitHub:
https://github.com/NonaSoftware/geneforge.

4 ACKNOWLEDGMENT

This work was supported by the National Science Foundation under Grant No. 2231864. 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] Choi, K., Medley, J. K., Cannistra, C., König, M., Smith, L., Stocking, K., and Sauro, H. M. Tellurium: A python based modeling and reproducibility platform for systems biology. *BioRxiv* (2016), 054601.

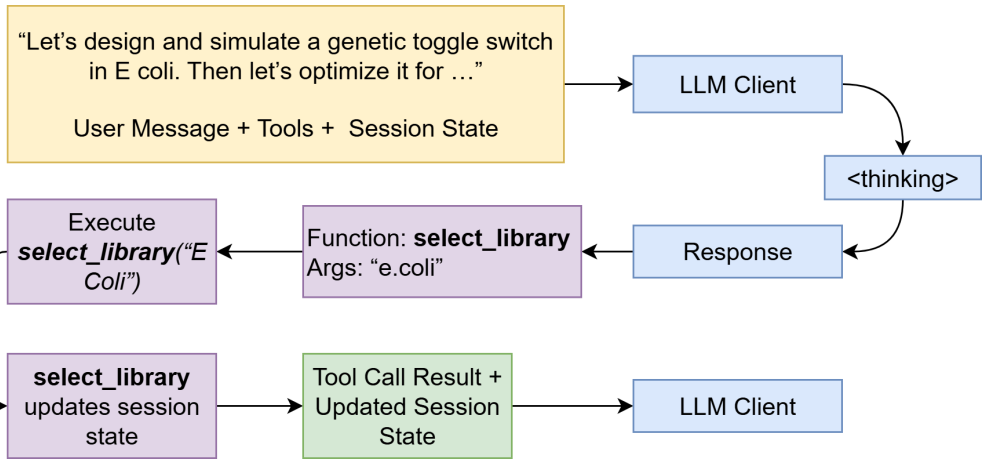


Figure 1: GeneForge Agent workflow for part selection.

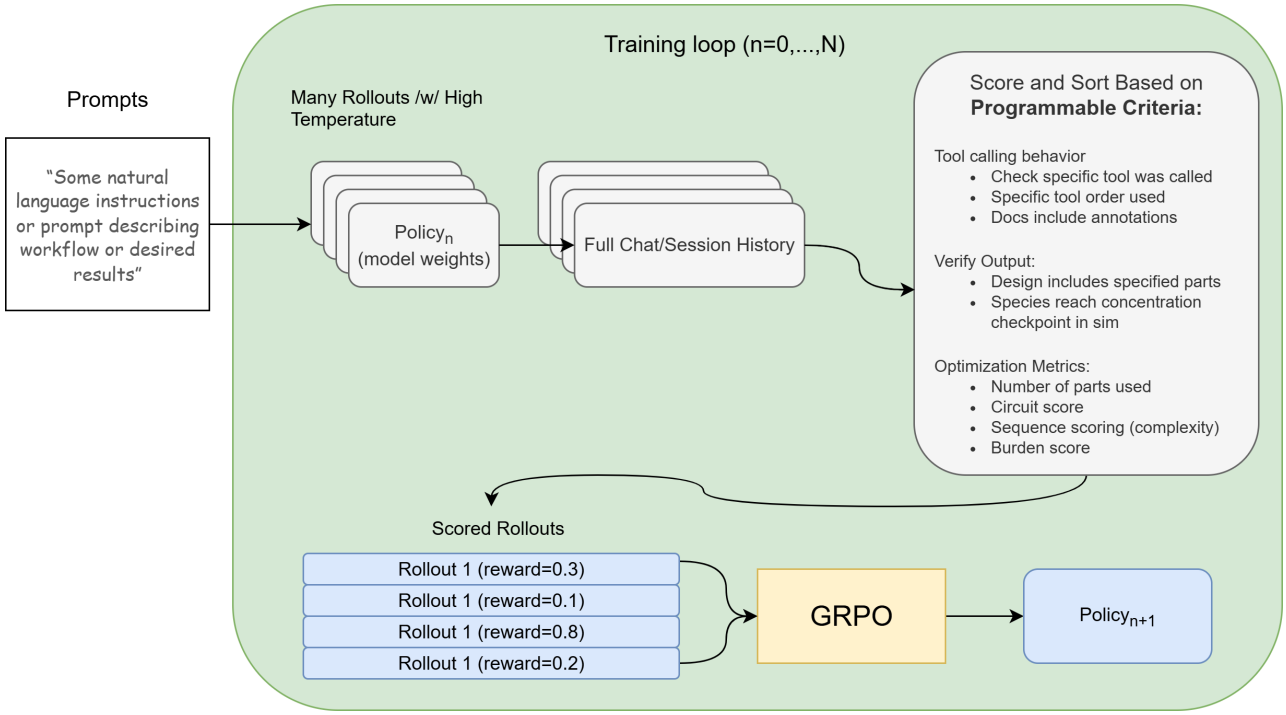


Figure 2: Workflow for fine-tuning with the GRPO method.

[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.

[3] Milo, R., Jorgensen, P., Moran, U., Weber, G., and Springer, M. Bionumbers—the database of key numbers in molecular and cell biology. *Nucleic acids research 38*, suppl_1 (2010), D750–D753.

[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.

[5] Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems 36* (2023), 53728–53741.

[6] Sane, S. Hybrid group relative policy optimization: A multi-sample approach to enhancing policy optimization. *arXiv preprint arXiv:2502.01652* (2025).

[7] Van Brempt, M., Clauwaert, J., Mey, F., Stock, M., Maertens, J., Waegeman, W., and De Mey, M. Predictive design of sigma factor-specific promoters. *Nature communications 11*, 1 (2020), 5822.

Beyond the boundary of university: Expanding Global Access to Synthetic Biology

Sudarshan Gc
Italian Institute of Technology, Italy
sudarshan.gc@iit.it

Prakriti Karki
TIGEM, Italy
p.karki@tigem.it

Synthetic biology, driven by the design–build–test–learn cycle or core engineering principles, is reshaping sectors from food and agriculture to textiles, medicine, diagnostics, and climate solutions. Despite rapid technological progress and the emergence of innovative tools, awareness and participation in synthetic biology remain concentrated in the Global North. Limited awareness in public or policy makers, a shortage of trained practitioners, and the high costs of conventional laboratory infrastructure continue to hinder its adoption in low- and middle-income countries. Democratizing synthetic biology requires reducing costs, enhancing accessibility and distribution, developing local expertise, and inspiring educators and students. Encouragingly, recent global initiatives, predominantly led by the Global North, are prioritizing these factors, offering models for building local synthetic biology ecosystems capable of solving context-specific challenges.

This review outlines the barriers to equitable participation in synthetic biology, highlights key initiatives addressing these gaps, and presents alternative low-cost, DIY approaches for research and education. It not only awares synthetic biology enthusiasts about the field and offers the idea to dive into this field but also offers practical guidance for early-career researchers, underfunded academic labs, community biolabs, synbio startups or even individuals in resource-limited settings seeking to establish functional laboratories with minimal equipment.

Keywords: synthetic biology, democratization, DIY biology, low-cost tools, resource-limited settings

Excel-SBOL Converter Version 2

Carolus Vitalis
University of Colorado Boulder
Boulder, Colorado
carolus.vitalis@colorado.edu

Taisiia Sherstiukova
University of Colorado Boulder
Boulder, Colorado
tash2960@colorado.edu

Gonzalo Vidal
University of Colorado Boulder
Boulder, Colorado
gonzalo.vidalpena@colorado.edu

Chris Myers
University of Colorado Boulder
Boulder, Colorado
chris.myers@colorado.edu

1 INTRODUCTION

Spreadsheets remain a cornerstone of biological data management, yet transforming DNA information from spreadsheets into standard-compliant documents, such as those using the *Synthetic Biology Open Language* (SBOL) [1], often requires bespoke scripts or manual intervention, impeding standardization. The original Excel-SBOL Converter [2] provided a first step toward bridging this gap, but encountered limitations in edge case handling, part-type coverage, and user onboarding. Here we present Version 2 of the converter, which delivers six major enhancements:

1. Expanded support for diverse DNA part categories.
2. Built-in validation rules in the Excel templates to catch common formatting errors upon entry.
3. SynBioHub authentication and duplicate-detection logic to guard against re-uploading existing parts.
4. A cross-platform graphical user interface (GUI) to guide users through the conversion workflow and visualize errors in real time.
5. Added functions to create complexes and interactions, such as protein production, promoter activation, and promoter repression.
6. Added functions for modules and components to support experimental metadata for the encoding of strains, media, and sample designs.

These improvements aim to lower the barrier to SBOL adoption, enforce consistency, and accelerate reproducible bio-design automation.

2 RESULTS

We refactored the original Python package to modularize conversion logic and implement comprehensive edge-case checks based on the SBOL 2 specification [1]. Concurrently, we developed macro-enabled Excel templates in Visual Basic for Applications (VBA) that enforce cell-level constraints (data types, mandatory fields) to minimize input mistakes.

To support hierarchical design representation, we implemented functions for creating modules and functional components directly from the Excel templates. Modules allow related ModuleDefinitions to be grouped into higher-level functional units, while functional components define typed roles and interactions of ComponentDefinitions within those modules. This enables complex genetic designs to be represented as nested compositions, with components reused across different constructs and experiments. To support simple pathway modeling, we implemented new functions that translate selected parts into SBOL Interaction and Complex objects, covering events such as protein production, promoter activation, and promoter repression, and embed them directly into the output document.

To make the Excel-SBOL Converter easier to use, we implemented a *graphical user interface* (GUI) (see

Figure 1), built in PySide6, which collects user information, reports errors, shows conversion progress, and allows users to sign in to SynBioHub to check existing libraries. SynBioHub [3] integration leverages its REST API for both user authentication and pre-upload queries to identify existing URIs and sequences.

We utilized the updated Excel-SBOL Converter in the DARPA TELLUS project to create a library of parts, as well as encode experimental metadata from tests performed on designs created from these parts. To support the part library, we created a new Excel template that includes

separate worksheets for each part type, since different types require different mandatory metadata. For example, promoters require activator or repressor links, whereas coding sequences must specify the protein they encode. Splitting these into individual tabs eliminates ambiguous columns and lets the validation routine flag omissions early, streamlining curation for wet-lab users. Part-type recognition was extended by mapping additional DNA component classes by expanding the part definition in the initialization sheet. In Excel-SBOL Converter Version 2, the part-specific worksheets can include headers that are strictly typed. All sheets share a core set of provenance columns: Part Id, Part Name, Part Description, Data Source, PubMed Id, DOI, Source Organism, Behavior, Length (bp), and Sequence, which map directly to SBOL displayId, name, description, and provenance terms. Each part type then adds its own mandatory fields. The promoter sheet holds “Activators” and “Repressors” (plus URI-resolved counterparts), the CDS sheet adds “Encodes for”, the Terminator and RBS sheets rely only on the core set [4], and

the Complex sheet captures component relations [5]. In total, the template covers nine component categories: Promoter, RBS, CDS, Terminator, Engineered Region, Protein, Complex, Small Molecule, and Other. All headers are enumerated in the column_definitions sheet and validated at runtime, a design that lowered template validation errors. In the DARPA TELLUS program, we used these sheets to create an SBOL representation for a part library comprising over 1246 distinct parts, and demonstrated a reduction in user input errors compared to the previous version.

During the DARPA TELLUS project, new Excel templates were developed to capture complete experimental context. New templates now encode host organism metadata (Chassis) with NCBI taxID identifiers, reagent information (Chemicals) such as inducers and antibiotics, and culture conditions (Media). Engineered organisms (Strains) link to both chassis and plasmids, while Supplements record in-sample concentrations of chemicals. The Sample Design sheet integrates strains, media, and supplements into a coherent definition of a sample composition, enabling dose–response or factor-based studies to be represented without redundant data entry. The Study and Assay template maintains a clear hierarchy, with assays capturing apparatus-level details and studies aggregating assays into publication-level units. Together, these templates enforce referential integrity, use controlled vocabularies for interoperability, and align with FAIR data principles [6]. They allow for a distributed description of an experiment from host strain to measurement, supporting direct SBOL export and integration with tools such as the Experimental Data Connector [7] for design–data linking and reproducible analysis.

Finally, the GUI enabled new users to generate SBOL files with little to no instruction.

In pilot runs, no duplicate uploads were observed, confirming that the SynBioHub query step blocks part IDs already present in the registry.

Our new interaction and complex creation features successfully modeled protein interactions in a single conversion pass, illustrating how spreadsheets can now serve not only as part catalogs but also as lightweight pathway builders.

Qualitative feedback from DARPA TELLUS participants highlights the value of inline error reporting and template enforcement in reducing back-and-forth with registry curators. By embedding standardization directly into spreadsheet templates, supporting seamless integration with repositories, and enabling basic interaction modeling, the Excel-SBOL Converter Version 2 advances bio-design automation best practices.

3 DISCUSSION

Remaining limitations include the absence of native SBOL 3 [8] export and limited support for non-DNA component types. Moving forward, we aim to deliver full SBOL 3 compatibility, enabling simpler representations and alignment with best practices in synthetic biology. In parallel, we plan to expand the experimental metadata templates toward compliance with the ISA (Investigation–Study–Assay) model [9], thereby standardizing how experimental context, protocols, and measurement data are structured and linked. ISA compliance will allow the Excel-SBOL Converter to interoperate seamlessly with a broader ecosystem of life science data management tools, supporting workflows that span from design through execution to data analysis. Additional planned features include ontology-driven part and metadata suggestions to improve data quality at entry, and a command-line interface to support high-throughput, automated pipelines. Collectively, these developments will extend the Excel-SBOL Converter from a genetic part conversion tool into a comprehensive bridge between design repositories, experimental metadata frameworks, and analysis platforms.

4 DATA AVAILABILITY

The updated converter with the new GUI, templates, and documentation is available on GitHub at: https://github.com/SynBioDex/Excel-to-SBOL.

5 ACKNOWLEDGMENTS

This work was supported by the Army Research Office under Cooperative Agreement Number W911NF-22-2-0210 and DARPA grant number HR0011-24-C-0423. 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] C. Madsen *et al.*, "Synthetic Biology Open Language (SBOL) Version 2.3," *J Integr Bioinform*, vol. 16, no. 2, Jun. 2019, doi: 10.1515/JIB-2019-0025.

[2] J. Mante, J. Abam, S. P. Samineni, I. M. Pötzsch, J. Beal, and C. J. Myers, "Excel-SBOL Converter: Creating SBOL from Excel Templates and Vice Versa," *ACS Synth Biol*, vol. 12, no. 1, pp. 340–346, Jan. 2023, doi: 10.1021/ACSSYNBIO.2C00521.

[3] J. A. McLaughlin *et al.*, "SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology," *ACS Synth Biol*, vol. 7, no. 2, pp. 682–688, Feb. 2018, doi: 10.1021/ACSSYNBIO.7B00403.

[4] K. Eilbeck *et al.*, "The Sequence Ontology: a tool for the unification of genome annotations," *Genome Biol*, vol. 6, no. 5, 2005, doi: 10.1186/GB-2005-6-5-R44.

[5] A. Bernasconi and M. Masseroli, "Biological and medical ontologies: Systems biology ontology (SBO)," *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, vol. 1–3, pp. 858–866, Jan. 2018, doi: 10.1016/B978-0-12-809633-8.20399-3.

[6] M. D. Wilkinson *et al.*, "The FAIR Guiding Principles for scientific data management and stewardship," *Sci Data*, vol. 3, Mar. 2016, doi: 10.1038/SDATA.2016.18.

[7] S. P. Samineni *et al.*, "Experimental Data Connector (XDC): Integrating the Capture of Experimental Data and Metadata Using Standard Formats and Digital Repositories," *ACS Synth Biol*, vol. 12, no. 4, pp. 1364–1370, Apr. 2023, doi: 10.1021/ACSSYNBIO.2C00669.

[8] L. Buecherl *et al.*, "Synthetic biology open language (SBOL) version 3.1.0," *J Integr Bioinform*, vol. 20, no. 1, Mar. 2023, doi: 10.1515/JIB-2022-0058.

[9] D. Johnson *et al.*, "ISA API: An open platform for interoperable life science experimental metadata," *Gigascience*, vol. 10, no. 9, pp. 1–13, Sep. 2021, doi: 10.1093/GIGASCIENCE/GIAB060.

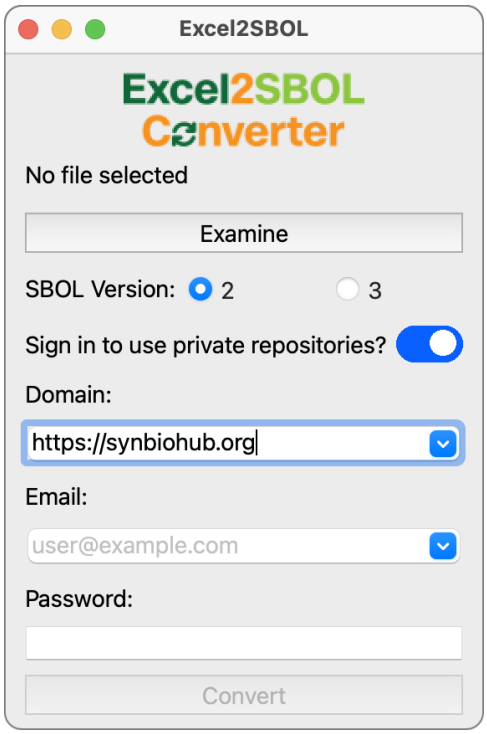


Figure 1. The new GUI lowers the entry barrier to the converter by eliminating command-line steps.

Towards a virtual GBM stem cell through integration of LLMs and graph analysis

Niloofar Arazkhani[1], Matei Zivanov[2], Difei Tang[1], Haomiao Luo[1], Nana Kwame Dwomoh[3], Brent Cochran[4], Natasa Miskov-Zivanov[1]

[1]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA

[2]Pittsburgh Allderdice High School, Pittsburgh, PA

[3]Brookings High School, Brookings, SD

[4]Department of Developmental, Molecular, and Chemical Biology, Tufts University School of Medicine, Boston, MA

Abstract. We compared the efficiency, effectiveness, and scalability of traditional natural language processing methods and recent large language models when extracting information from research papers in the glioblastoma multiforme context. Our results suggest that traditional NLP is more efficient, providing significantly larger output, while LLMs output more precise and more detailed information about intracellular interactions to better guide mechanistic modeling and creation of virtual GBM stem cells.

1 INTRODUCTION

Glioblastoma multiforme (GBM) is an aggressive form of brain tumor with a median survival of 10 months. Typical treatments include surgery, chemotherapy and radiotherapy. However, these treatments do not remove stem cells, which later differentiate again and result in tumor relapse. Stem cell targeted therapies, in combination with chemotherapy and radiotherapy have been suggested as a more potent approach [1].

Computational mechanistic models based on expert knowledge offer not only predictions of promising treatments, but also explanations and understanding of why the treatments may work and if there are any potential side effects. Entities in mechanistic models are usually genes, proteins, chemicals, and biological processes (e.g., apoptosis, proliferation, DNA damage). Biological events such as phosphorylation, binding, transcription are modeled as entity interactions and with sate update rules. Mechanistic model simulations provide information about the dynamic behavior – changes in entity states over time.

Much of the information that can be used to develop a mechanistic model is available in the published literature. In this study, we compare the utility of traditional natural language processing (NLP) approaches with more recent large language model (LLM)-driven approaches for automatically creating a mechanistic model of GBM stem cell (virtual cell).

2 METHODOLOGY

Our methodology for extracting events and comparing the traditional NLP-driven approach (INDRA [2]) with an LLM-driven approach (LLaMa 3 [3]) is illustrated in Figure 1 (a). The first step of our workflow is selecting published papers that serve as a knowledge source. We conducted four searches, Q1-Q4, detailed in Table 1. To collect paper PMC IDs, we searched PubMed with three queries (Q1, Q3, Q4) and used a repository of GBM-relevant literature assembled by an expert in SciWheel [4] (Q2).

The next step is the extraction of interactions. INDRA uses the list of paper IDs as input and it outputs extracted events from these papers in a structured JSON format. The information extracted by INDRA is converted to a structured list of interactions in the BioRECIPE format [5] with a converter script [6]. The BioRECIPE format includes the information about the event, the regulator and regulated entity, as well as the context and provenance information. To use LLaMa 3, we first created several scripts that retrieve papers using paper IDs and then convert them to plain text format. We provided these text files to LLaMa and collected its output through an API access. We also designed a prompt, shown in Figure 1 (b), and with the few-shot prompting approach we instructed LLaMa to output the collected information in the BioRECIPE format.

Finally, the interaction lists are filtered with FLUTE [7] to retain only interactions highly supported by interaction databases. Although this step increases the confidence in the interaction list at the output of FLUTE, it may remove more recent and novel observations that are not yet included in databases, and which have a potential to improve the model. Therefore, we conducted the analysis of the interaction lists both before and after the FLUTE filtering step.

3 RESULTS

3.1 Literature retrieval

The search query results are listed in Table 1. The number of papers found as relevant with Q1 is much larger than all the other lists. The Q3 paper list size is similar to Q2, which was manually collected by an expert. Interestingly, the number of available texts for processing in the list Q2 was significantly smaller than for Q3, indicating that PubMed may be a more reliable source for ranking literature. Adding only one word

to the Q4 query significantly reduced the number of retrieved publications.

3.2 Extraction of interaction lists

As can be seen from Table 1, INDRA extracted approximately 18.4 interactions per paper, while LLaMa extracted only 0.8 interactions per paper. Additionally, INDRA extracted 2,005 interactions per hour, whereas LLaMa extracted only 200 interactions per hour. This highlights INDRA's efficiency in both speed and interaction extraction compared to LLaMa 3. However, while LlaMa 3 outputs significantly smaller number of interactions than INDRA, it provides more details about interactions, populating more element and interaction attributes with values in the BioRECIPE format.

3.3 Analysis of assembled networks

We assembled and analyzed the networks generated by each extraction method individually (INDRA and LlaMa 3) and when integrated with the interaction filtering tool (INDRA+FLUTE, LLaMa 3+FLUTE), for Q1-Q4, compared them with a manually created and curated GBM model [8] and summarized the results in Figure 2.

Network visualizations in Figure 2(a) and numbers in Figure 2(b) show that networks obtained with INDRA have longer diameter than those output by LlaMa 3 in all cases except Q1, while still significantly shorter than the manually curated model, even though the model is much smaller than the Q1 networks. Longer network diameter indicates more connected interactions and thus, longer signaling pathways.

After using FLUTE with the output of INDRA and LlaMa 3, we observed that INDRA+FLUTE integration reduces the number of connected components to just 2% of the original INDRA output, whereas the LLaMa+FLUTE integration retains 12.5% of the original LLAMA output. This stronger filtering effect on INDRA's network suggests that LLaMa 3 produces higher confidence interactions that are more precise and relevant.

3.4 Insights for modeling

We conducted analysis of several node and edge network features for all obtained networks. Some of these results are shown in Figure 3. All networks exhibit a scale-free structure, that is, only few nodes have high values for in-degree and out-degree metrics. Interestingly, this matches the typical structure of biological networks, where highly connected nodes, known as hubs, play a crucial role [9]. For INDRA and LlaMa 3 outputs, these results likely stem from their many disconnected components and the varying frequency of elements in the literature.

Further, stress and betweenness centrality follow a similar pattern, with most nodes having low values and only a small subset with high values. High-stress nodes tend to influence strongly a larger portion of the network. In Figure 3, we listed elements with high stress for several INDRA and LLaMa 3 networks. Interestingly, most of these elements (STAT3, TP53, EGFR, ERK, SOX2, HIF1A, MYC, PTEN, MTOR, AKT, apoptosis, and NOTCH1), align with expert biological knowledge and with the key elements in the manually curated model. Additionally, several new elements (CNTN3, CTNNB1, FOXM1) that are not in the curated model are identified through this graph analysis approach. These results further highlight the potential of our approach for automated model assembly and refinement.

4 Conclusion

Our study demonstrates that traditional NLP approaches and LLMs offer complementary advantages in efficiency and precision, suggesting that a combined approach could enhance information extraction and assembly for mechanistic modeling. INDRA is very efficient in extracting interactions, however, it often outputs interaction with scarce information, while LLaMa 3 provides more details about element and interaction attributes. To expand our results, we plan to investigate parallelization of processing with LLaMa to study the entire corpus of 60K GBM publications found on PubMed. We will also explore the enhancements of prompting strategies.

5 REFERENCES

[1] S. Mohammed, M. Dinesan, and T. Ajayakumar, "Survival and quality of life analysis in glioblastoma multiforme with adjuvant chemoradiotherapy: a retrospective study," Rep Pract Oncol Radiother, vol. 27, no. 6, pp. 1026-1036, 2022.

[2] B. M. Gyori, J. A. Bachman, K. Subramanian, J. L. Muhlich, L. Galescu, and P. K. Sorger, "From word models to executable models of signaling networks using automated assembly," Molecular Systems Biology, vol. 13, no. 11, p. 954, 2017.

[3] L. T. A. a. Meta, "The Llama 3 Herd of Models," arXiv preprint, 2024.

[4] Sciwheel. "Sciwheel – A Reference Management and Research Collaboration Tool." (accessed 2025).

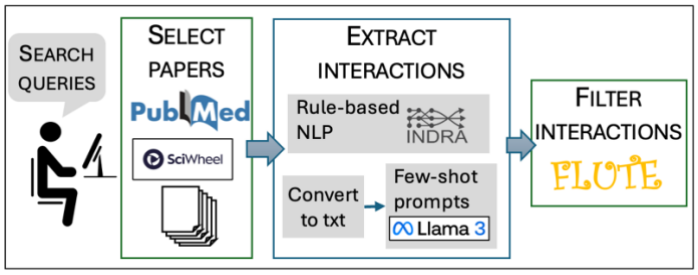
[5] E. Holtzapple et al., "The BioRECIPE Knowledge Representation Format," ACS Synth Biol, vol. 13, no. 8, pp. 2621-2624, Aug 16 2024.

[6] M.-Z. Lab. "BioRECIPE Documentation." (accessed 2025).

[7] E. Holtzapple, C. A. Telmer, and N. Miskov-Zivanov, "FLUTE: Fast and reliable knowledge retrieval from biomedical literature," Database (Oxford), vol. 2020, Jan 1 2020.

[8] B. C. Emilee Holtzapple, Natasa Miskov-Zivanov, "Automated verification, assembly, and extension of GBM stem cell network model with knowledge from literature and data," bioRxiv, 2021.

[9] A. Butchy, N. Arazkhani, C. Telmer, and N. Miskov-Zivanov, "Automating Knowledge-Driven Model Recommendation: Methodology, Evaluation, and Key Challenges," EEE Transactions on Computational Biology and Bioinformatics 2025.



(a)

[text:]

Loss of the S6K1-mediated feedback loop resulting from mTORC1 inhibition enhances PDK1-mediated phosphorylation of Akt at Thr308. Consequently, when suboptimal doses of mTOR catalytic inhibitors were used, the residual mTORC2 activity towards Ser473 potently activated Akt.

#split#

Biological Regulator:
Name: mTORc2;
Type: protein;
Subtype: kinase;
HGNC Symbol: MTOR;
Database: UniProt;
ID: P42345;
Compartment Name: cytoplasm;
Compartment ID: GO:0005737

Biological Regulated:
Name: AKT;
Type: protein;
Subtype: kinase;
HGNC Symbol: AKT1;
Database: Uniprot; ID: P31749;
Compartment Name: cytoplasm;
Compartment ID: GO:0005737

Interaction:
Sign: positive;
Connection Type: direct;
Mechanism: phosphorylation;
Site: Ser473;
Cell Line: empty;
Cell Type: GBM cell;
Tissue Type: brain;
Organism: human

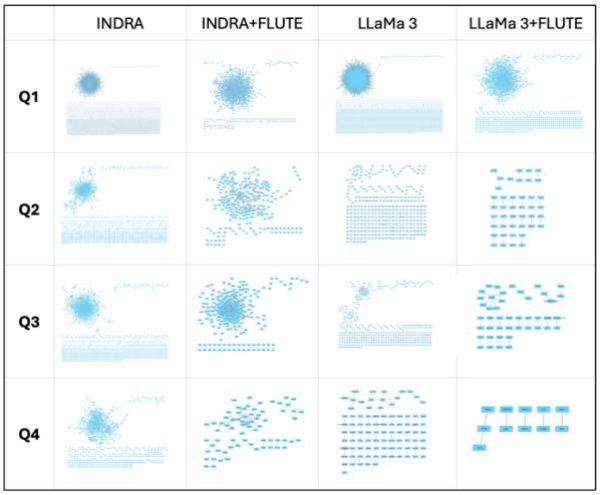
(b)

Figure 1 *(a) Interaction extraction and evaluation workflow. (b) An example prompt used with LLaMa 3: a sentence from a paper and the expected output (structured interaction with BioRECIPE attributes).*

Table 1*: Summary of literature retrieval with queries Q1-Q4.*

Query	Information source	Description	Publication year	Paper IDs	Full text not found	Papers processed		Interactions extracted		Runtime [h]	
						INDRA	Llama 3	INDRA	Llama 3	INDRA	Llama 3
Q1	PubMed	"glioblastoma multiforme stem cells"	2005-2009	91	30	61	85	605	35,247	~0.5	~120
			2010-2014	714	169	545	593	6,539		~4	
			2015-2019	1,450	303	1,147	432	17,900		~8	
			2020-2025	1,791	774	1,017	854	24,515		~10	
Q2	SciWheel	Relevant papers selected by expert	2004-2024	449	167	202	140	5,297	207	~2.5	~24
Q3	PubMed	""glioblastoma multiforme" and (EGFR or PDGFR or VEGFR)"	2005-2025	448	51	336	396	577	290	~2.7	~30
Q4	PubMed	""glioblastoma multiforme" and stem and (EGFR or PDGFR or VEGFR)"	2009-2025	83	4	67	74	1,111	50	~0.5	~5

More focused search for Q1 was used as the shorter “glioblastoma multiforme” query returned ~60,000 papers, which would take LLaMa 3 impractically long to process in our current setup. Not all papers have freely accessible full texts, and therefore, we selected for processing only those papers with available full texts.



(a)

Query	Network name	Number of nodes	Number of edges	Network diameter	Network density	Connected components
	GBM Model	411	555	30	0.003	7
Q1	INDRA	28,960	49,534	4	0.000	5,005
	INDRA + FLUTE	1,143	3,612	11	0.002	48
	LLaMa 3	27,794	35,247	3	0.000	4,720
	LLaMa 3 + FLUTE	1475	2,092	11	0.001	225
Q2	INDRA	4,750	5,297	15	0.000	1,154
	INDRA + FLUTE	254	420	9	0.005	23
	LLaMa 3	354	207	4	0.002	160
	LLaMa 3 + FLUTE	42	22	1	0.013	20
Q3	INDRA	2,850	4,345	12	0.000	505
	INDRA + FLUTE	239	472	8	0.006	21
	LLaMa 3	425	290	5	0.002	154
	LLaMa 3 + FLUTE	44	28	2	0.014	17
Q4	INDRA	847	1,111	10	0.001	136
	INDRA + FLUTE	69	81	5	0.014	13
	LLaMa 3	89	50	2	0.006	43
	LLaMa 3 + FLUTE	11	6	1	0.055	5

(b)

Figure 2. *(a) Visualization of networks obtained from INDRA, and LlaMa 3 (and with FLUTE) for Q1-Q4. (b) Graph metrics obtained for a manually curated GBM model and for the networks illustrated on the left.*

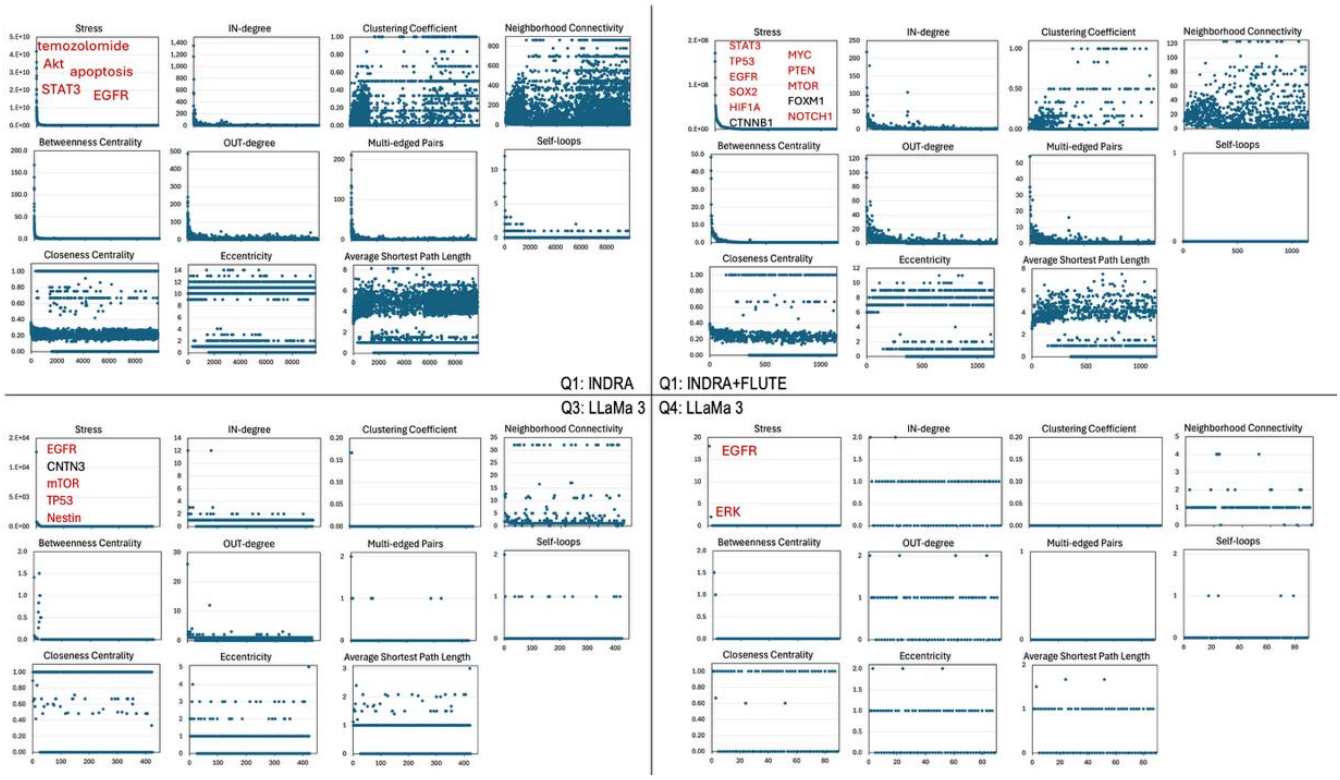


Figure 3*. Graph metrics of individual nodes in four of the 16 obtained networks: INDRA and INDRA+FLUTE output for Q1, LLaMa 3 output for Q3 and Q4. Definitions: Stress - number of shortest paths passing through a particular node; IN-degree – number of incoming edges; Clustering Coefficient - neighbor connectivity density; Neighborhood Connectivity - average degree of neighbors); Betweenness Centrality - normalized measure of a node’s role as a bridge in shortest paths; OUT-degree - outgoing edges; Multi-edged Pairs - parallel edges between nodes; Self-loops - autoregulatory edges;, Closeness Centrality - global proximity to others; Eccentricity - maximum distance to others; and Average Shortest Path Length - mean distance across nodes.*

BuildPlanner: A tool for connecting the Design and Build stages of the DBTL cycle

Ryan Greer
University of Colorado Boulder
Boulder, United States of America
Ryan.Greer@colorado.edu

Kerem Gurkan
University of Colorado Boulder
Boulder, United States of America
Kerem.Gurkan@colorado.edu

Derick Sayavong
University of Colorado Boulder
Boulder, United States of America
Derick.Sayavong@colorado.edu

Carolus Vitalis
University of Colorado Boulder
Boulder, Colorado
Carolus.Vitalis@colorado.edu

Chris Myers
University of Colorado Boulder
Boulder, United States of America
Chris.Myers@colorado.edu

Gonzalo Vidal
University of Colorado Boulder
Boulder, United States of America
Gonzalo.VidalPena@colorado.edu

1 INTRODUCTION

Synthetic biology is an interdisciplinary field that aims at engineering biology. As an engineering discipline, it has the *Design, Build, Test, and Learn* (DBTL) cycle at its core. The DBTL cycle is a framework that provides a systematic and iterative approach to the engineering of biological systems. Researchers have implemented ad hoc methods for iterating the DBTL cycle and often do not use standards for data sharing, making it difficult to understand and reproduce results [3]. The synthetic biology community developed the *Synthetic Biology Open Language* (SBOL) to standardize biological designs [1, 2]. Over time, the SBOL community updated the standard to represent the entire DBTL cycle and developed an ecosystem of tools to automate data encoding. In particular, a best practice for the representation of build plans was developed to provide an interface for software tools between the Design and Build stages [7]. This paper presents BuildPlanner, a Python package that will take abstract designs and create build plans encoded in SBOL.

2 RESULTS

BuildPlanner is a Python package that implements SBOL best practices for standardized representation of parts and assembly for build planning. BuildPlanner begins with an abstract design, and then identifies plasmids with the parts needed to assemble it, and creates a plan that includes all the interactions and intermediate states. The input abstract design is a composition of genetic parts (i.e., promoter, RBS, CDS, and terminator) in a backbone (see Figure 1). Then, our package searches for plasmids containing such parts on SynBioHub and retrieves a combination that would work to build the abstract design. The output assembly plan is an SBOL file containing the assembly interactions (i.e., digestion, ligation, provenance), as well as intermediate states (i.e., part extract, open backbone) (see Figure 2). The assembly plan represents information that bridges designs and build plans.

Encoding Golden Gate Assembly: The current version of BuildPlanner supports the construction of abstract designs using Golden Gate assembly plans. BuildPlanner uses PyDNA to simulate the digestion of plasmids [4]. Next, ligation combines all extracted parts by overhang complementarity. This step generates a list of potential DNA constructs while systematically recording the ontology of each input, transformation, and product in an SBOL2-compliant assembly [2].

Linking Assembly to Robotic Workflows: The resulting build plan is rich in metadata, connecting original parts to their composite constructs. This representation serves as input for PUDU, a software tool that automates liquid handling in synthetic biology workflows [8]. PUDU identifies all composite parts in the backbone, traces their origins to the input plasmids, and registers them as reagents and creates an automated assembly protocol.

Integration into SynBioSuite: SynBioSuite is a genetic design and simulation platform with a graphical user interface (GUI) that enables users to construct genetic circuits through a drag-and-drop interface [5]. In its second version, SynBioSuite expanded beyond design to support the entire DBTL cycle, incorporating BuildPlanner to bridge the design and build stages. Within this workflow, BuildPlanner takes an abstract design generated in SBOLCanvas [6] (which is integrated into SynBioSuite), creates a corresponding build plan, and passes it to PUDU, which produces an automated assembly protocol. This integration gives BuildPlanner a user-friendly GUI, enabling a smooth transition from computational design to physical assembly, and improving both reproducibility and efficiency in genetic construct fabrication (see Figure 3).

3 DISCUSSION

BuildPlanner currently provides a standardized bridge between genetic design and lab assembly but is limited to Golden Gate methods and SBOL2. Extending support to

widely used techniques like Gibson Assembly and support of SBOL3 [1] would broaden its applicability, allowing users to select the best strategy based on design, part availability, and experimental goals. Leveraging SBOL-encoded build plans as a direct interface to DNA synthesis companies could automate order preparation, validate construct feasibility, and reduce human error in transitioning from *in silico* designs to physical DNA. Additionally, integrating with laboratory automation frameworks such as PyLabRobot [9] would enable execution of complex workflows across multiple robotic platforms without extensive custom coding. Together, these enhancements would make BuildPlanner a more versatile, interoperable, and industry-ready tool, supporting diverse assembly methods and automation platforms.

4 ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation, under Grant Nos. 2231864, the Army Research Office under Cooperative Agreement Number W911NF-22-2-0210, and DARPA grant number HR0011-24-C-0423. 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. This project was originally created with the support of Asad Malik, Chris Krenz, and Doug Densmore during the NonaWorks 2024 session to implement their manufacturing canvas at DAMP LABS.

5 ADDITIONAL RESOURCES

You can find more information regarding BuildPlanner on our Read the Docs documentation `https://sbol2build.readthedocs.io/en/latest/Tutorials.html`. You can also view the package code, raise issues, and contribute to the packages development on our open source GitHub repository `https://github.com/MyersResearchGroup/SBOL2Build`.

REFERENCES

[1] Buecherl, L., Mitchell, T., Scott-Brown, J., Vaidyanathan, P., Vidal, G., Baig, H., Bartley, B., Beal, J., Crowther, M., Fontanarrosa, P., et al. Synthetic biology open language (sbol) version 3.1. 0. *Journal of Integrative Bioinformatics 20*, 1 (2023), 20220058.

[2] Madsen, C., Goñi Moreno, A., P, U., Palchick, Z., Roehner, N., Atallah, C., Bartley, B., Choi, K., Cox, R. S., Gorochowski, T., et al. Synthetic biology open language (sbol) version 2.3. *Journal of integrative bioinformatics 16*, 2 (2019), 20190025.

[3] Mante, J., and Myers, C. J. Advancing reuse of genetic parts: progress and remaining challenges. *nature communications 14*, 1 (2023), 2953.

[4] Pereira, F., Azevedo, F., Carvalho, Â., Ribeiro, G. F., Budde, M. W., and Johansson, B. Pydna: a simulation and documentation tool for dna assembly strategies using python. *BMC bioinformatics 16*, 1 (2015), 142.

[5] Sents, Z., Stoughton, T. E., Buecherl, L., Thomas, P. J., Fontanarrosa, P., and Myers, C. J. Synbiosuite: a tool for improving the workflow for genetic design and modeling. *ACS Synthetic Biology 12*, 3 (2023), 892–897.

[6] Terry, L., Earl, J., Thayer, S., Bridge, S., and Myers, C. J. Sbolcanvas: a visual editor for genetic designs. *ACS Synthetic Biology 10*, 7 (2021), 1792–1796.

[7] Vidal, G., Vitalis, C., and Guillén, J. Standardized golden gate assembly metadata representation using sbol. In *Golden Gate Cloning: Methods and Protocols*. Springer, 2024, pp. 89–104.

[8] Vidal, G., Vitalis, C., and Rudge, T. Pudu: Simple liquid handling robot control for synthetic biology workflows. In *International Workshop on BioDesign Automation (IWBDA)* (2023).

[9] Wierenga, R. P., Golas, S. M., Ho, W., Coley, C. W., and Esvelt, K. M. Pylabrobot: an open-source, hardware-agnostic interface for liquid-handling robots and accessories. *Device 1*, 4 (2023).

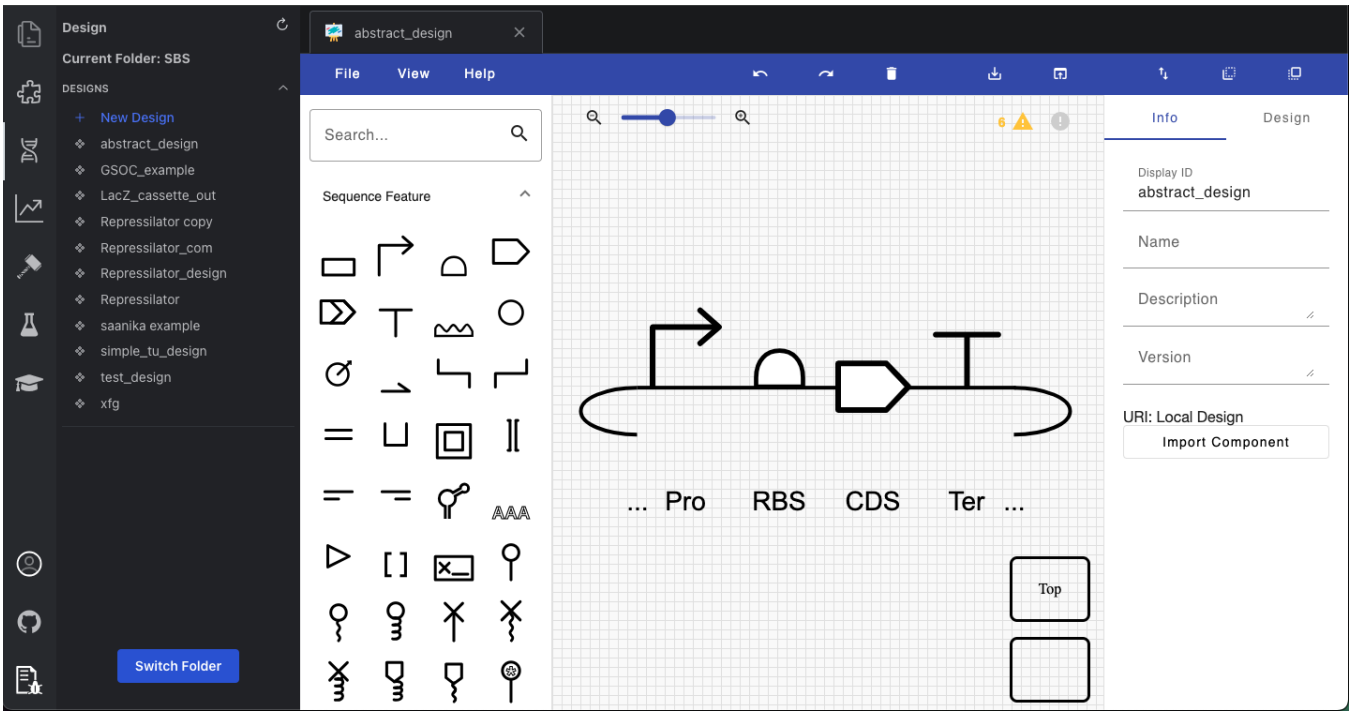


Figure 1: Diagram of an abstract design. SynBioSuite GUI's design tab screen shot. The design tab uses SBOLCanvas for genetic design. The abstract design include the parts (Pro, RBS, CDS, Ter) and backbone needed for the assembly.

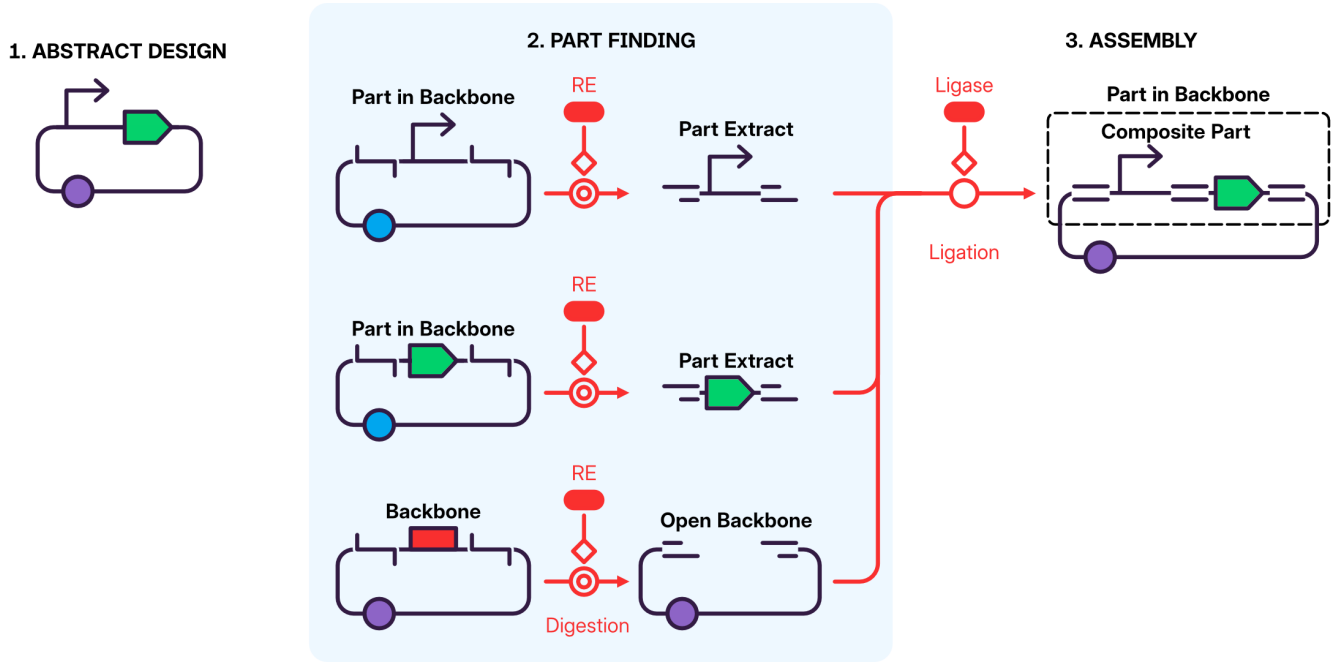


Figure 2: Standard build plan SBOL Visual diagram. 1. The abstract design is a simple SBOL file, where parts (i.e., promoter and CDS) and backbone are specified and can be imported from SynBioHub. 2. BuildPlanner finds parts in backbone from collections in SynBioHub, then it simulates a golden gate assembly and retrieves all the possible ways to implement the abstract design. On the top left two parts in backbone are represented, one containing a promoter and other containing a green CDS. On the bottom left a backbone is represented containing a red CDS. These three plasmids are digested using a restriction enzyme, producing part extracts from the parts in backbone and an open backbone from the backbone. 3. The three digestion products are then ligated using a ligase, producing a part in backbone where the part is a composite part of the two parts extract that can be assembled using the selected collection of plasmids in SynBioHub.

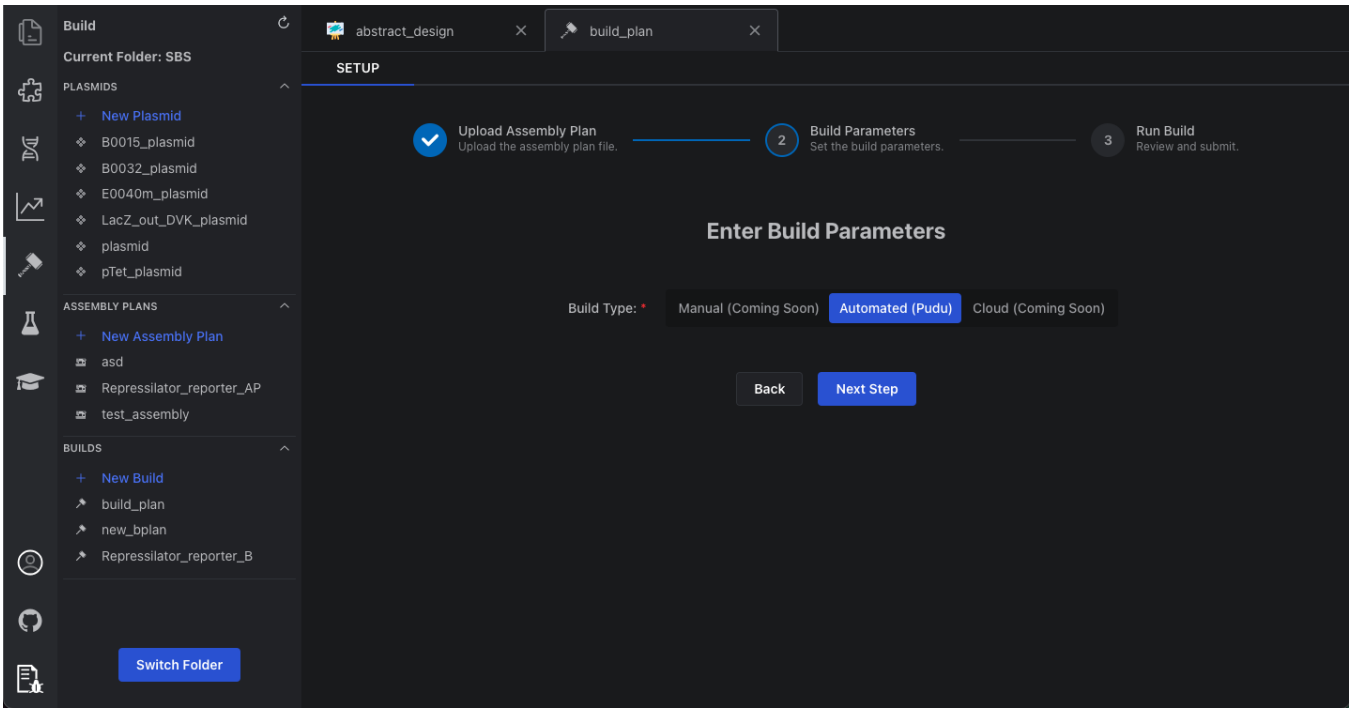


Figure 3: Connection using BuildPlanner. SynBioSuite GUI's build tab screen shot. Assembly plans can be dragged and dropped into a build to create an automated protocol for the assembly of the abstract design.

Digital Microfluidics for DNA Data Storage and Bio-Computation

arman.hajizadeh@gmail.com
Pars Human Gene
Tehran, Tehran

1 INTRODUCTION

DNA, with unmatched density and stability, constitutes an energy-favorable and environmentally sustainable medium for data storage, encoding text, images, and video into nucleotide sequences retrievable through nanopore readout. Extending beyond storage, biocomputing—an emergent frontier of molecular computation—where binary transitions enable binomial equation solving [1], CRISPR CPUs [2], and molecular artificial neural networks[3] embodied in living machines [4], bioelectronic and quantum-inspired biological systems, and lab-on-chip platforms [5], and extending even to implanted–artificial intestine robots [6].

As microfluidics anchors lab-on-chip technology, Conventional paradigms – microchannel architectures with intricate microvalve networks and syringe-driven hydraulics, microtiter well plates coupled to liquid handler robots suffering from jetting instabilities and volumetric metering artifacts reminiscent of inkjet precision constraints, and classical pipet-based liquid transfer constrained by stochastic irreproducibility and intensive manual throughput demands – could not fully resolve the bottlenecks of automation in the synthetic biology field [7].

Digital microfluidics (DMF), through the actuation of discrete droplet microreactors on addressable electrode arrays, circumvents these constraints by enabling programmable, dynamically multiplexable, valve-free nanoliter–microliter liquid operations with reduced biofouling, minimal consumables, and seamless integration of high-fidelity automated phenotypic assays and precision interrogation of unicellular dynamics, synthetic protocell constructs, and engineered intercellular communication networks.

In DMF platforms, droplets can be actuated by magnetic gradients, holographic optical trapping, acoustic fields, and—most prominently— by electric-field-driven electrowetting. In this work, we investigate droplet transport, elongation, and splitting as rudimentary operations toward fully automated DMF devices, while further analyzing biomolecular adsorption phenomena—such as protein and nucleic acid surface interactions—through PCR as an information-coping operation, benchmarking our real-time optical readout system against a commercial thermocycler.

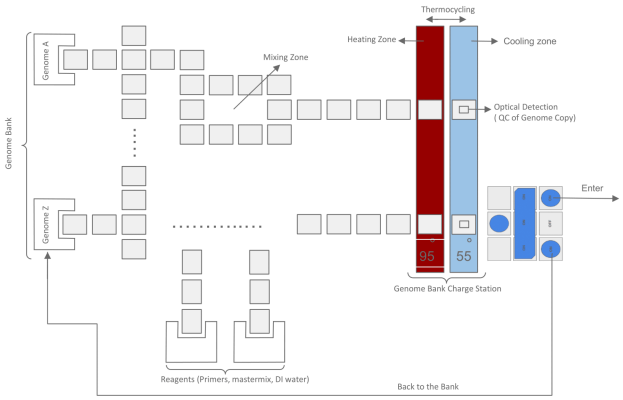


Figure 1: Schematic Platform for a PCR-on-chip device

2 MATERIALS AND METHOD

A digital microfluidic (DMF) chip was fabricated using standard printed circuit board (PCB) manufacturing with patterned zigzag electrodes ((2 mm)) separated by a 0.1 mm gap. A dielectric layer was prepared by spin-coating melted parafilm to a thickness of approximately 5 μm, with spin speed and duration empirically adjusted to account for the viscosity of molten parafilm. To render the surface hydrophobic, the electrode array was coated with FluoroPel® PFC1601V (Cytonix, USA) using a multi-stage spin coating, resulting in a $\sim$ 100 nm layer, followed by thermal curing at 180°C for 10 min. Droplets containing PCR reagents were introduced into an oil medium consisting of propylene glycol and silicone oil (5 cSt). To further suppress nonspecific adsorption and improve droplet stability, Pluronic® F127 was first dissolved at the appropriate ratio in *N*, *N*-dimethylformamide (DMF), and subsequently added to the droplet formulation at a final concentration of 0.05 % (w/v). For validation of real-time amplification, the DMF platform was benchmarked against a commercial real-time thermocycler, the *Mic qPCR Cycler* (BioMolecular Systems, Queensland, Australia).

3 RESULTS & DISCUSSION

Using the fabricated DMF platform designated for PCR (Figure 1), we successfully conducted three-stage thermal cycling

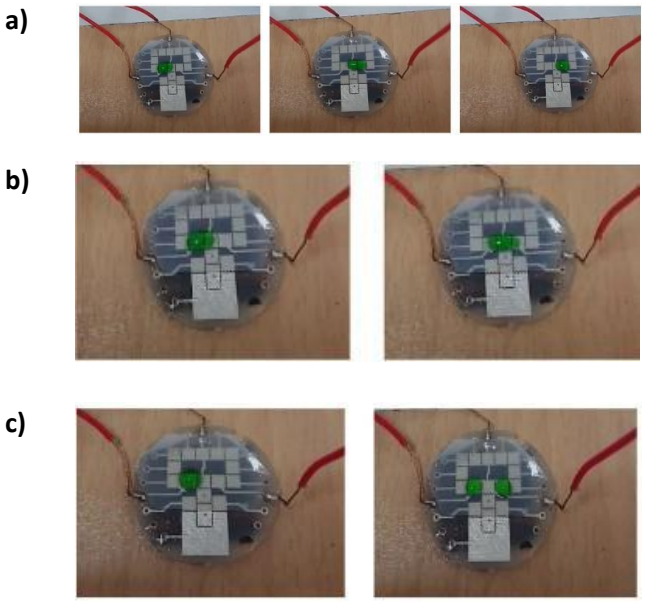


Figure 2: Experimental investigation on the droplet operations: a) Motion, b) Elongation, c) Splitting

with heating and cooling elements regulated by an Arduino-controlled thermocouple to switch cyclically between hot and cold domains. In initial trials, a single droplet was actuated at 100 VDC, exhibiting smooth motion across the electrode array. Repeatability tests over 20 consecutive switching cycles confirmed stable transport without observable errors or barriers (Figure 2a). By activating three adjacent electrodes simultaneously, controlled droplet elongation was achieved (Figure 2b), and subsequent high-precision stretching enabled droplet splitting into nearly equal daughter volumes. One daughter droplet exceeded the other by less than 0.05 in relative volume (Fig. 2c). Comparative PCR amplification data (Figure 3) demonstrated no detrimental effect on DNA yield or amplification efficiency between the DMF device and the commercial control system.

The optimized droplet formulation minimized evaporation, suppressed biomolecular surface adsorption—including proteins and whole cells—and enhanced droplet mobility across the electrode array. Iterative testing of surfactant type and concentration revealed this to be a favorable operational parameter. Future optimization may include substituting the parafilm dielectric layer with materials such as SU-8 or PVDF to reduce dielectric thickness, as well as engineering bilayer dielectric stacks to balance charge retention with water resistivity, thereby improving device longevity. Furthermore, implementation of a top ITO-coated glass plate with a spin-coated hydrophobic layer represents a potential strategy to decrease actuation voltages, reduce biofouling, and further minimize contamination of biological reagents.

As is evident from the convergence of DMF and DNA amplification methodologies, electric-field actuation in digital microfluidics can unify PCR-on-chip amplification, automated synthetic biology pipelines, electroporation-driven gene editing [8], DNA sequencing library preparation [9], Proteomics sample preperation [10] and GelMA-based core-shell droplet configurations for quorum-sensing cell sorting [11, 12], while simultaneously supporting IPTG-inducible [13], chemically, and DNA-based multi-input–gated biological logic circuits across bacterial communities –DPGA in droplet and droplet-based DPGA –, from wildtype isolates to engineered mutant strains [14]. Systematic validation of these operations lays the foundation for a fully automated DNA data storage and molecular biocomputation platform.

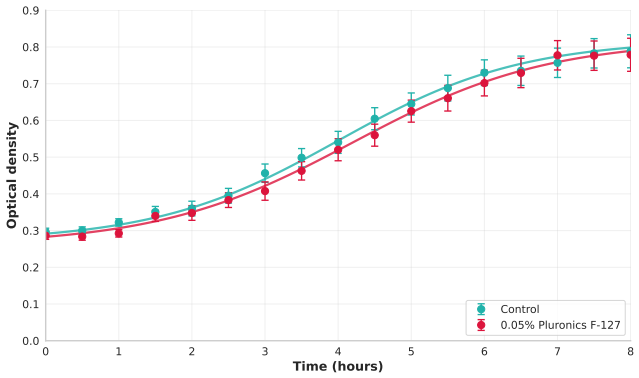


Figure 3: A PCR amplification curve obtained with (red) and without (cyan) 0.05% Pluronic F-127.

REFERENCES

[1] Hui Lv, Nuli Xie, Mingqiang Li, Mingkai Dong, Chenyun Sun, Qian Zhang, Lei Zhao, Jiang Li, Xiaolei Zuo, Haibo Chen, et al. Dna-based programmable gate arrays for general-purpose dna computing. *Nature*, 622(7982):292–300, 2023.

[2] Hyojin Kim, Daniel Bojar, and Martin Fussenegger. A crispr/cas9-based central processing unit to program complex logic computation in human cells. *Proceedings of the National Academy of Sciences*, 116 (15):7214–7219, 2019.

[3] Xiewei Xiong, Tong Zhu, Yun Zhu, Mengyao Cao, Jin Xiao, Li Li, Fei Wang, Chunhai Fan, and Hao Pei. Molecular convolutional neural networks with dna regulatory circuits. *Nature Machine Intelligence*, 4 (7):625–635, 2022.

[4] Fahim Farzadfard and Timothy K Lu. Genomically encoded analog memory with precise in vivo dna writing in living cell populations. *Science*, 346(6211):1256272, 2014.

[5] Sharon Newman, Ashley P Stephenson, Max Willsey, Bichlien H Nguyen, Christopher N Takahashi, Karin Strauss, and Luis Ceze. High density dna data storage library via dehydration with digital microfluidic retrieval. *Nature communications*, 10(1):1706, 2019.

[6] Yichao Shi and James H Pikul. Achieving animal endurance in robots through advanced energy storage. *Science Robotics*, 10(101):eadr6125, 2025.

[7] Mathieu C Husser, Philippe QN Vo, Hugo Sinha, Fatemeh Ahmadi, and Steve CC Shih. An automated induction microfluidics system for synthetic biology. *ACS Synthetic Biology*, 7(3):933–944, 2018.

[8] Hugo Sinha, Angela BV Quach, Philippe QN Vo, and Steve CC Shih. An automated microfluidic gene-editing platform for deciphering cancer genes. *Lab on a Chip*, 18(15):2300–2312, 2018.

[9] Qingyu Ruan, Weidong Ruan, Xiaoye Lin, Yang Wang, Fenxiang Zou, Leiji Zhou, Zhi Zhu, and Chaoyong Yang. Digital-wgs: Automated, highly efficient whole-genome sequencing of single cells by digital microfluidics. *Science advances*, 6(50):eabd6454, 2020.

[10] Max K Steinbach, Jan Leipert, Theo Matzanke, and Andreas Tholey. Digital microfluidics for sample preparation in low-input proteomics. *Small Methods*, 9(1):2400495, 2025.

[11] BA Nestor, E Samiei, R Samanipour, A Gupta, A Van den Berg, M Diaz de Leon Derby, Z Wang, H Rezaei Nejad, K Kim, and M Hoorfar. Digital microfluidic platform for dielectrophoretic patterning of cells encapsulated in hydrogel droplets. *RSC Advances*, 6(62):57409–57416, 2016.

[12] Zongliang Guo, Fenggang Li, Hang Li, Menglei Zhao, Haobing Liu, Haopu Wang, Hanqi Hu, Rongxin Fu, Yao Lu, Siyi Hu, et al. Deep learning-assisted label-free parallel cell sorting with digital microfluidics. *Advanced Science*, 12(1):2408353, 2025.

[13] Alex JH Fedorec, Neythen J Treloar, Ke Yan Wen, Linda Dekker, Qing Hsuan Ong, Gabija Jurkeviciute, Enbo Lyu, Jack W Rutter, Kathleen JY Zhang, Luca Rosa, et al. Emergent digital bio-computation through spatial diffusion and engineered bacteria. *Nature Communications*, 15(1):4896, 2024.

[14] Jai P Padmakumar, Jessica J Sun, William Cho, Yangruirui Zhou, Christopher Krenz, Woo Zhong Han, Douglas Densmore, Eduardo D Sontag, and Christopher A Voigt. Partitioning of a 2-bit hash function across 66 communicating cells. *Nature Chemical Biology*, 21(2): 268–279, 2025.