EasyChair Preprint

№ 9031

Rule-Based Generation of Synthetic Genetic Circuits

Daisuke Kiga, Kazuteru Miyazaki, Shoya Yasuda, Ritsuki Hamada, Sota Okuda, Ryoji Sekine, Naoki Kodama and Masayuki Yamamura

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

October 8, 2022

Rule-based generation of synthetic genetic circuits

Daisuke Kiga[1], Kazuteru Miyazaki[2], Shoya Yasuda[3], Ritsuki Hamada[1], Sota Okuda[1], Ryoji Sekine[3], Naoki Kodama[4], Masayuki Yamamura[3]

[1]Waseda University, Tokyo, Japan, [2]National Institution for Academic Degrees and Quality Enhancement of Higher Education, Tokyo, Japan, [3]Tokyo Institute of Technology Kanagawa, Japan, [4]Meiji University Kanagawa, Japan

kiga@waseda.jp, teru@niad.ac.jp, my@c.titech.ac.jp

Introduction

Similar to the expandability of natural biological systems, that of synthetic biological systems is derived from the huge combinatorial search space of biological components, such as protein coding sequences and regulatory sequences [1]. Due to this huge space, adequate design strategies are required for the implementation of synthetic genetic circuits in cells.

One design strategy for genetic circuits is a combination of sub-circuits. Recent progress in automated computational design has achieved multi-layered logic gates [2, 3]. Another direction of computational design can be reliance on expert knowledge. Indeed, even manual combinations of sub-circuits have allowed the implementation of prescribed cellular behavior [4, 5].

To develop a support tool for genetic-circuit design by biologists, here, we sought to combine inference machine and deep learning to generate and screen candidates of synthetic genetic circuits, respectively. Once an adequate rulebase is prepared, a logic programming language such as Prolog allows the designed cellular behavior to be broken down into combinations of rules, each understandable by a biologist. Simultaneously, each combination can indicate a genetic network topology from which published tools can estimate adequate parameters and suggest genetic parts [6, 7]. Although inference engines can potentially cause combinatorial explosions, using machine learning for candidate screening before the numerical simulation can circumvent this problem.

Results and Discussion

1. Combinations of rules provide multiple strategies for a prescribed cellular behavior

Logic programs can design prescribed cellular behaviors appearing from a certain combination of rules. In this work for circuit design, we will prepare a rulebase on general biological knowledge that is

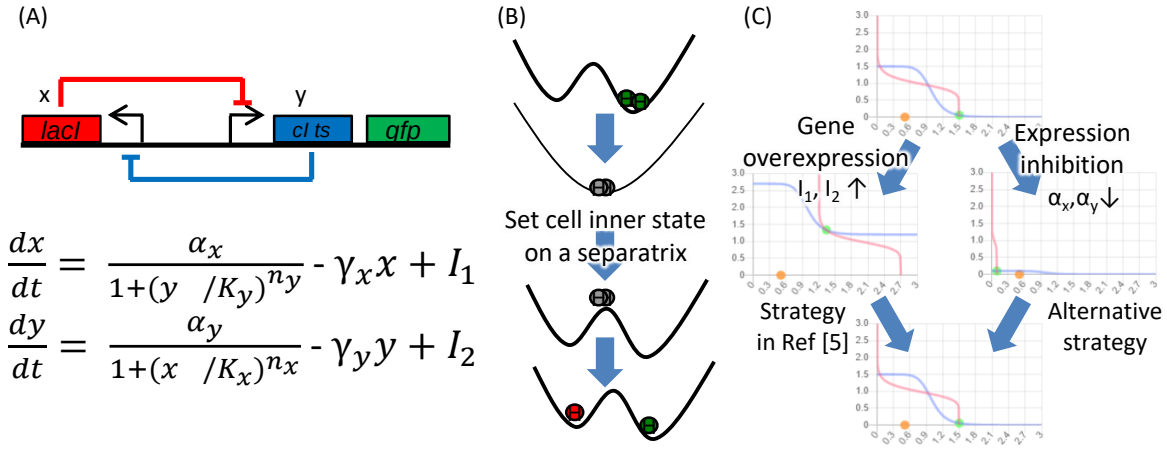

Figure 1: Multiple strategies for genetic reprogramming can be generated by a combination of rules for the Inference Engine. (A) Genetic toggle switch structure and ODEs for each repressor of the toggle switch. (B) A process for genetic reprogramming. (C) Phase space and nullclines for the toggle switch system.

The research was funded by JST, CREST Grant Number JPMJCR21N4, Japan

independent of specific circuits. A user can then provide a specification as the goal for the logic program. After successive dissections into smaller subgoals, these subgoals indicate the usage of small general genetic circuits, such as a toggle switch for bistability (Figure 1A) or a gene overexpression system.

In the case of our genetic reprogramming of a toggle switch (Figure 1) [5], one of the internal subgoals is setting an inner state of cells on a separatrix of the potential landscape (Figure 1B). In our study, this setting was achieved by gene overexpression, which increased I1 and I2 in the equations in Figure 1. These increases make a parallel shift of nullclines to decrease the number of nullcline intersections (Figure 1C). Another way to achieve this setting was the inhibition of the expression of both repressors. From a logic programming viewpoint, variations of the circuits for the same prescribed behavior were generated because two rules shared the same head: "setting an inner state of cells on a separatrix". A strong point of the inference engine is that any user can easily add new rules to the rulebase, because the logical inference engine substantially includes consistency checks among rules.

2. Generation of genetic networks by Inference engine and Screening of the generated network

Although combinatorial explosion is a known weak point for generation by an inference machine, we think that recent developments in machine learning will allow appropriate screening of the genetic circuits generated as candidates.

Figure 2 shows our design process. Step 1: Generation of candidates. As described in the previous section, an inference engine generates network candidates by combinations of rules. Step 2: Screening by comparison of features between a designed circuit and each of the circuits in a database. By using ML, we will evaluate the similarity to a known genetic network, and decide the searching priority by reinforced learning. Step 3: Numerical calculations for various sets of parameters for the topology of a circuit. This step requires the highest computational cost in the whole process. Thus, pruning or ranking of candidates at Step 2 is important. Step 4: Corresponding to the parameters of the screened candidate, appropriate biological components are selected from the database. At Steps 3 and 4, we can use published design tools if they can be integrated with a Prolog compiler.

3. Semi-automated collection of articles for synthetic genetic circuits

Towards the construction of the genetic circuit database, we started a collection of network topology figures in synthetic biology articles. For the semi-automated collection of articles, we used machine learning of network topology figures of related studies. By manual classification, we provided positive example papers, each with at least one figure for the topology of the synthetic gene network. Papers that lacked any network topology figure were also manually classified as negative example papers. As positive examples, we chose 361 figures from 70 positive example papers in ACS Synthetic Biology. To avoid bias, we must use similar numbers of negative and positive papers. As negative examples, we thus used 505 figures from 85 ACS Synthetic Biology papers not related to genetic circuits. After training, 46 genetic circuit articles from other journals, as well as 185 negative example papers, were evaluated (Table 1). Further developments will allow more accurate classification.

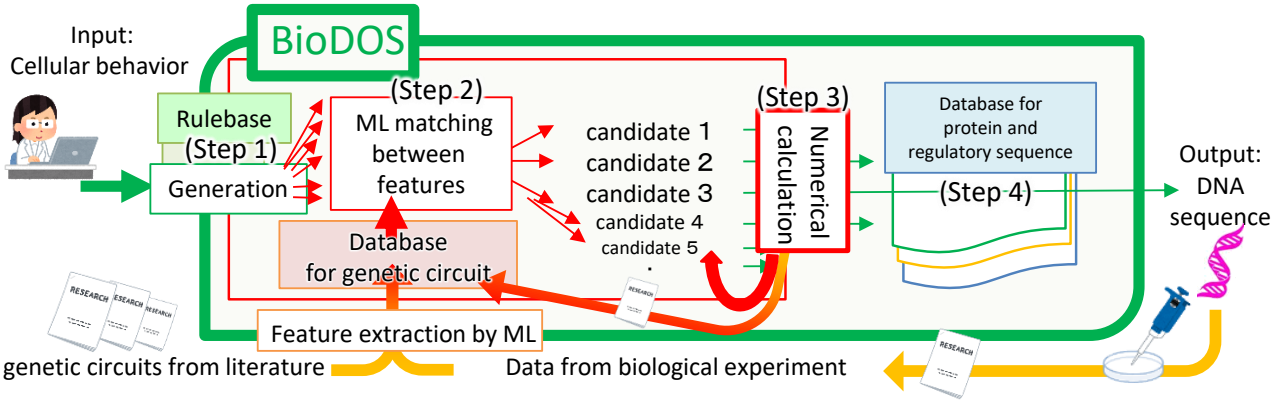

Figure 2: Generation and screening process of candidate circuits for a cellular behavior.

Collected literature will be used not only for screening the candidate circuits generated by the inference engine, but also for providing information to researchers who will expand the rulebase of the inference engine.

	percentage correctly classified
Positive example papers	61.8
negative example papers	63.3

Table 1: Classification of genetic circuit papers

Future Directions

Using Prolog, we started a description of the rules to generate circuit topologies and parameters for cellular behavior, as shown in Figure 1. The accumulation of such rules will allow researchers with biology backgrounds to write new rules for the rule base and to implement new circuits showing what life could potentially be.

REFERENCES

[1] A Casas, et al., R Kitney. Removing the Bottleneck: Introducing cMatch - A Lightweight Tool for Construct-Matching in Synthetic Biology. Front Bioeng Biotechnol. 9 (2022), 785131.

[2] T Jones, et al., D Densmore. Genetic circuit design automation with Cello 2.0. Nat Protoc. 17 (2022) 171097-1113.

[3] H Tas, et al., V de Lorenzo. Automated design and implementation of a NOR gate in Pseudomonas putida. Synth Biol (Oxf). 6 (2021) 6ysab024.

[4] R Sekine, et al., and D Kiga. Tunable synthetic phenotypic diversification on Waddington's landscape through autonomous signaling. PNAS. 108 (2011) 17969-17973.

[5] K Ishimatsu, et al., and D Kiga. General Applicability of Synthetic Gene-Overexpression for Cell-Type Ratio Control via Reprogramming. ACS Synth. Biol., 3, (2013) 638–644.

[6] Y Boada, et al., A Vignoni. Multi-objective optimization framework to obtain model-based guidelines for tuning biological synthetic devices: an adaptive network case. BMC Syst Biol. 10 (2016) 27.

[7] N Dalchau, et al., A Phillips. Towards the rational design of synthetic cells with prescribed population dynamics. J R Soc Interface. 9 (2012) 2883-98.