Modeling *Bio*BLESS: The Gillespie's Algorithm

Lingyuan Ji

iGEM Wiki

Major Models Used in Gene Circuit Simulation

In order to investigate the mechanism of a gene circuit system, one choose a mathematical model of the system. The mainstream modeling method and their main features are listed below:

1. the ordinary differential equation method (the ODE method)
    - deterministic model
    - Hill function as an assumption
    - many empirical parameters
    - easy to implement and run fast
2. the stochastic differential equation method (the SDE method)
    - stochastic model
    - Hill function as an assumption
    - many empirical parameters
3. the partial differential equation method (the PDE method)
    - deterministic model
    - taking the spacial distribution of species into consideration
    - relatively slow
    - space-time dependent PDEs are not easy to solve
4. the chemical master function method (the CMF method)
    - stochastic model
    - taking all possible state of the system into consideration
    - using PDEs to calculate the state possibility evolution
    - numerically untreatable due to the high storage need
5. the Gillespie algorithm method (the GA method)

- stochastic model
- only one evolution generated in one session
- few parameters and few assumptions
- easy to implement but relatively slow

The ODE method with Hill function as an assumption is the most popular model in iGEM team's modeling part. This deterministic model may lead to a system stability problem because the micro-world in cells is intrinsically stochastic. Internal perturbation and parameter inaccuracy is inevitable so the computer checked circuit maybe unusable in laboratory.

In order to solve the above problem, our team choose the Gillespie Algorithm in *Bio*BLESS. With the help of this algorithm, the stochastic behavior of the system can be easily demonstrated and fully examined.

Gillespie Algorithm

THE *only* assumption of Gillespie algorithm is:

> For a given set of molecules as reactants of a given reaction, the possibility of the reaction do happen in the following dt time interval is, in first order, proportional to dt, with the proportional constants c_νs.

In order to make some formal deduction, we use the following set of variable to describe the state of the system:

$$X_i(t) = \text{the molecules number of chemical spice } i \text{ in the system}$$

and describe the interaction between the spices using the following chemical reaction set:

$$l_1^{(\mu)} S_1 + l_2^{(\mu)} S_2 + \cdots + l_n^{(\mu)} S_n \to r_1^{(\mu)} S_1 + r_2^{(\mu)} S_2 + \cdots + r_n^{(\mu)} S_n.$$

With the above preparation, the deduction can be made as follow.

Suppose that the system is at time t, we define a probability function that describe the system's next chemical reaction event:

$$\begin{aligned} P(\tau,\mu)\mathrm{d}\tau = & \text{the probability, in first order, that the next reaction} \\ & \text{take place in time interval } (t+\tau, t+\tau+\mathrm{d}\tau). \end{aligned}$$

Under this definition, we can obtain the analytic form of $P(\tau,\mu)$ by using the fundamental hypothesis.

Firstly, we find the possible molecules combination number of each reaction, which is obviously given in combination number as follow:

$$h_\mu = \prod_{i=1}^{n} \binom{X_i(t)}{l_i^{(\mu)}}, \ \mu = 1, 2, \cdots, m.$$

These h_νs are time dependent, and will be used later.

Secondly, we define the following probability function, as a lemma to the final answer.

$$\begin{aligned}P_0(\tau) = &\text{the probability that there is no reaction}\\ &\text{take place in time interval } (t, t+\tau)\end{aligned}$$

To obtain the formula of $P_0(\tau)$, one just need to divide time interval $(t, t+\tau)$ into N independent equal part, and use the fundamental hypothesis to ensure that there is no reaction in the infinitesimal interval:

$$\begin{aligned}P_0(\tau) &= \lim_{N\to\infty}\left\{\prod_{\nu=1}^{m}[1 - h_\nu c_\nu \epsilon + o(\epsilon)]\right\}^N\\ &= \lim_{N\to\infty}\left[1 - \sum_{\nu=1}^{m} h_\nu c_\nu \epsilon + o(\epsilon)\right]^N\\ &= \lim_{N\to\infty}\left[1 - \sum_{\nu=1}^{m} h_\nu c_\nu \frac{\tau}{N} + o\left(\frac{\tau}{N}\right)\right]^N\\ &= \exp\left(-\sum_{\nu=1}^{m} h_\nu c_\nu \tau\right)\end{aligned}$$

Thirdly, we use Conditional Probability Formula to construct the analytic form of $P(\tau, \mu)$: the probability that the next reaction is reaction μ in time interval $(t+\tau, t+\tau+\mathrm{d}\tau)$ equals the $P_0(\tau)$ times the probability that the reaction in $\mathrm{d}\tau$ is reaction μ. That is to say:

$$P(\tau, \mu) = h_\mu c_\mu \exp\left(-\sum_{\nu=1}^{m} h_\nu c_\nu \tau\right).$$

Up to now, we have obtained the form of $P(\tau, \mu)$ just from out first fundamental hypothesis. Our only left task is to generate the random `float` τ and random `int` μ from their joint distribution $P(\tau, \mu)$.

At the end, we show how to generate (τ, μ) from $(0, 1)$ uniformly distributed random number. In order to do this, the marginal distribution of τ is needed:

$$P_1(\tau) = \left(\sum_{\sigma=1}^{m} h_\sigma c_\sigma\right)\exp\left(-\sum_{\nu=1}^{m} h_\nu c_\nu \tau\right).$$

Then, we use the Conditional Probability Formula again to find the distribution of μ when τ is generated.

$$P_2(\mu|\tau) = \frac{P(\tau, \mu)}{P_1(\tau)} = \frac{h_\mu v_\mu}{\sum_{\sigma=1}^{m} h_\sigma c_\sigma}$$

This is exactly what we want.

Finally, we can describe the whole algorithm:

1. simulation preparation
    (a) allocate storage space and write initial molecules count
    (b) input the proportional constants mentioned above
    (c) input the simulation time range
2. event-driving simulation
    (a) calculate the next reaction time t_{next} using:
    $$t_{\text{next}} = t_{\text{cur}} + \frac{1}{\sum_{\nu=1}^{m} h_\nu c_\nu} \ln \frac{1}{r_1}$$
    (b) decide the next reaction type μ using:
    $$\sum_{\nu=1}^{\mu-1} h_\nu c_\nu \leq r_2 \sum_{\nu=1}^{m} h_\nu c_\nu \leq \sum_{\nu=1}^{\mu} h_\nu c_\nu$$
    (c) use the above (t_{next}, ν) to modify the current time and molecules counter
    (d) add current record to records set

    The c_νs is the proportional constants of reaction ν mentioned above, and h_νs is the count of possible reactant molecules sets of reaction ν. r_1 and r_2 are independently generated random numbers uniformly distributed on $(0, 1)$.
3. termination and output
    (a) if t_{current} exceeds stop time, stop the calculation
    (b) plot the records set data

GA in BioBLESS

In BioBLESS, our team implements a stochastic simulator using GA in `Python`. This simulator is integrated in the `/backend/BioBLESS/biosys/reaction_system` module in our project. It can analyse the gene circuit constructed in BioBLESS `Logic` part, and automatically translate it into GA-executable primitive reaction. One can modify every parameters in GA (or just leave them in default) and easily trigger a stochastic simulation with a simple click.

One blame on GA is that it seems to be slow when the gene circuit is complex. In BioBLESS, a gene circuit contain 3–5 logic gates may take one minute to simulate. This is a little long. To ensure that average users can have a quick access to simulation results using common parameters in BioBLESS, our team create a `backend/BioBLESS/biosys/bio_system_cache` utility. When a simulation request is sent to BioBLESS server, this module hash the input file first. If this input file was simulated before, this module will return the previous result stored in the database. Besides the `cache` module, the complete user-defined simulation using GA can also be done in BioBLESS anyway.