Communications in Information and Systems
Volume 18, Number 4, 209–228, 2018

Role of combinatorial complexity in genetic networks

Weihua Geng[†] and Xin Yang

A common motif found in genetic networks is the formation of large complexes. One difficulty in modeling this motif is the large number of possible intermediate complexes that can form. For instance, if a complex could contain up to 10 different proteins, 2^{10} possible intermediate complexes can form. Keeping track of all complexes is difficult and often ignored in mathematical models.

Here we present an algorithm to code ordinary differential equations (ODEs) to model genetic networks with combinatorial complexity. In these routines, the *general* binding rules, which counts for the majority of the reactions, are implemented automatically, thus the users only need to code a few *specific* reaction rules. Using this algorithm, we find that the behavior of these models depends greatly on the specific rules of complex formation. Through simulating three generic models for complex formation, we find that these models show widely different timescales, distribution of intermediate states, and ability to promote oscillations within feedback loops. These results provide tools for the incorporation of combinatorial complexity of genetic networks, and show how this incorporation may be vital to accurately predict the network dynamics.

1. Introduction

Proteins within cells barely act alone. Instead, they form complexes to become active, regulable, modifiable and destroyable. Unfortunately, this presents a great difficulty for the computational biologists. As the number of proteins increases, the possible complexes grow exponentially. For this reason, combinatorial complexity is almost always ignored in mathematical models for biochemical reaction networks by making simplified assumptions, which may or may not be appropriate.

Recent modelings are attempted to explore this complexity. One of the examples is a mathematical model for the circadian (24-hour) clock within

[†]Weihua Geng is the corresponding author of this paper.

mammalian cells [1], where the possible combinations of the various proteins (e.g. PER1, PER2, CRY1, CRY2 and kinases) were directly simulated. The simulation software packages also appeared in assisting the study of combinatorial complexity. The implementations fall into two approaches.

The first approach is to automatically create the full reaction network, and then specify interaction patterns and binding constraints for macromolecular complexes, which are then used to automatically generate reaction equations. The typical example is BioNetGen program [2], whose algorithm has been included in Virtual Cell [3]. BioNetGen includes the consideration of combinatorial complexity and specializes in modeling dynamics of a signal transduction system, including enzymatic activities, post-translational modifications and interactions of the domains of signaling molecules.

The second approach simulates the reaction networks without ever explicitly creating it, and a typical example is Moleculizer program [4]. Moleculizer starts from a description of monometric proteins and specification for reactions such as binding, unbinding and others. It incorporates new complexes and reactions only when needed, and therefore is reduced to much smaller reaction networks. It also contains both deterministic formulas for species presented in terms of real number concentrations, and stochastic equations for integer species amount changed by probabilistic rules.

In the present model, the reactions in generic networks are composed of the *general* binding reactions between molecules, and *specific* reactions such as transcription, translation, transportation, phosphorylation and so on. The specific reactions, due to their specialty and relatively small amount in number, are usually handled individually. The general binding procedure, which often involves a large number of participating molecules and entails combinatorial complexity, needs to be treated automatically to generate coupled differential equations, using reactions constants as parameters.

To explore combinatorial complexity in genetic networks, we developed a methodology to automatically generate the equations for a model given the biological assumptions about association and dissociation of participated species. This method is fast and flexible. Including another protein in the complex can be accounted for just a few keystrokes, even though it may double the number of differential equations. Considering several generic examples, we find that the dynamics of these networks are exquisitely sensitive to the rules of complex formation and display behaviors that would not have been seen without direct simulation of all possible complexes.

We will provide details about the study of role of combinatorial complexity in genetic networks with particular emphasis on the differences of these models in simulating different combinatorics mechanism and the method to generate system of ODEs efficiently. A fixed dimension example of these mechanisms has been reported in our previous work [5], which calls attention in many related work [6–17]. Current work is a generalization of methodology in *any* dimension with more technical details in numerical implementation. We will introduce the involved three models in the next section, followed by algorithms and implementation details, and then the numerical simulation results and their discussion. The article ends with concluding remarks.

2. Models

2.1. Feedback control mechanism

We are interested in finding the periodic solution (e.g. $\mathbf{y}(t) = \mathbf{y}(t+T)$) for a given genetic network. For the resulting system of ODEs, the existence of limit cycle indicates the existence of periodic solution. However, the theory of finding limit cycle is only available for a two-dimensional system with the Poincarè-Bendixson theorem such that on a phase portrait an unstable spiral or node is found inside a confine set Ω (i.e. $\mathbf{n} \cdot \dot{\mathbf{y}}(t) < 0$ for $\mathbf{y}$ on $\partial\Omega$, where $\mathbf{n}$ is the normal vector) [18]. For this reason, we seek period solution for our high-dimensional system by intuition from the negative feedback control, assisted with numerical simulation.

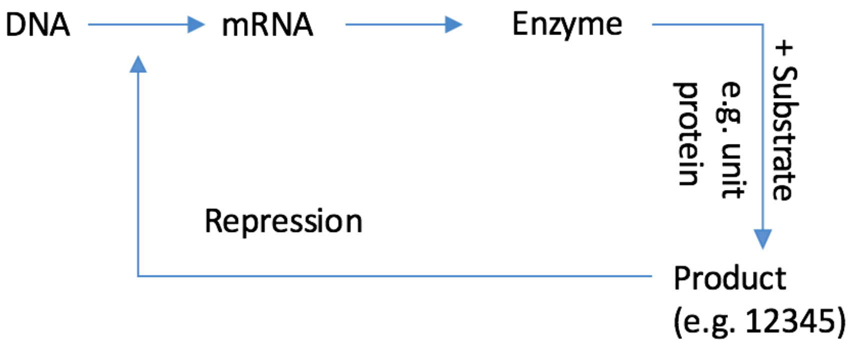


Figure 1: Goodwin's negative feedback mechanism: the product, which is produced from the enzyme-substrate binding, represses the transcription of DNA to mRNA to repress the synthesis of the enzyme.

We take the Goodwin's model [19], which is shown in Fig. 1, as an example. The product, which is produced by binding the substrate to an enzyme synthesized under the direction of the messenger RNA (mRNA) using genetic information transcribed from the DNA, represses the enzymes of their own synthesis by repressing the transcription of the molecule DNA to messenger RNA (mRNA). The system of ODEs, which simulates this model, is given as [18]

$$\frac{dM}{dt} = \frac{V}{D + P^n} - aM \tag{1}$$

$$\frac{dE}{dt} = bM - cE \tag{2}$$

$$\frac{dP}{dt} = dE - eP \tag{3}$$

where M, E, P are concentration of the mRNA, enzyme, and product respectively, a, c, e are the corresponding degradation constants, b and d are rate constants. E and P are created and degraded by first order kinetics. The creation of M is inhibited by the product P and is degraded according to first-order kinetics. To see this, note the function $f(P) = \frac{V}{D+P^n}$ is a decreasing function of P, where n is the Hill coefficient, indicating a negative feedback. In other words, the increment of P causes the decrement of $f(P)$, which eventually makes the whole term $f(P) - aM$ negative, thus the decrement of M. The decrement of M makes the aM term smaller and smaller, which could lead to the positive values of the whole $f(P) - aM$ term for again the increment of M.

Next section will focus on the study of combinatorial complexity, i.e. the formation for the final product involving many arbitrary intermediate steps, which is much more complicated than the Goodwin's model in Eqs. (1–3). However, the way that the final product represses the initial reactants is similar to the way that the product represses the translation in Goodwin's model.

2.2. Combinatorial complexity

Assuming that a system initially has only free unit proteins (the smallest unit in the network), which could bind to each other to form complexes, we present three models with different binding mechanisms. It worths noting that these three models are inspired by previous work of Forger and Peskin [1] in the study of feedback system involving DNA and a series of clock proteins such as CLK, BMAL, PER1, PER2, CRY1, CRY2, CKI-ϵ, CKI-δ, CKI-inhibitor, etc. The interaction of these proteins suggested the following three mechanisms. In practice, there are many other possible mechanisms

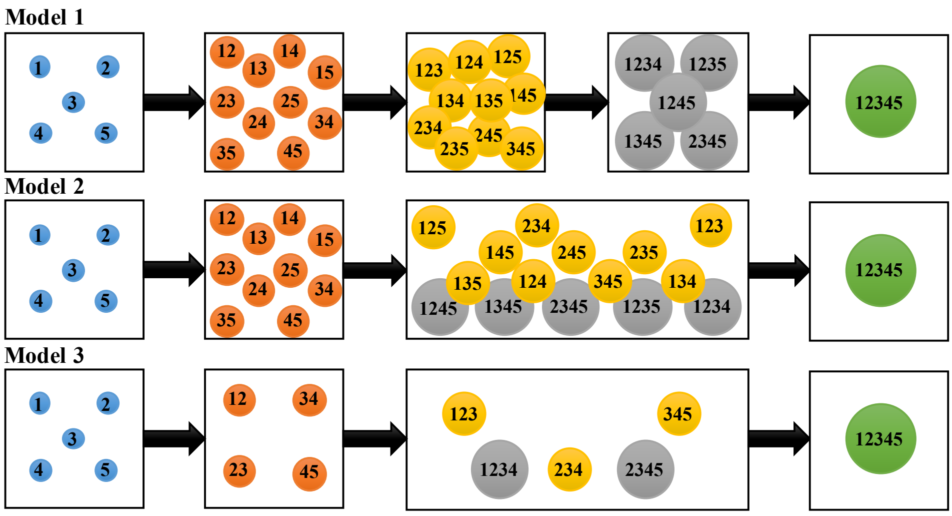


Figure 2: Illustration of three generic models of complex formation.

for general genetic networks. These three models describe the association of proteins step by step to eventually form a *final* complex, which will in turn, repress the production of initial unit proteins. Figure 2 illustrates the mechanism of the three models, showing only the new complexes at each step.

Model 1: a unit protein binds to a protein/complex

In Model 1, only a unit protein can bind to another unit protein to form a complex or bind to an existing complex that does not contain the present protein to form a new complex. The following are the assumptions related to the reactions:

1) A new protein binds to the complex at each step;
2) Initially there are only N types of unit proteins;
3) Eventually a *final* complex containing one piece of each unit protein is formed as the inhibitor/repressor to the unit proteins.

Model 2: protein/complex containing different unit proteins bind

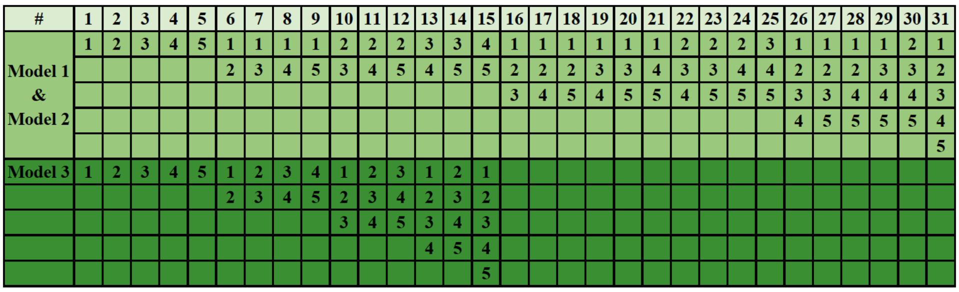
In Model 2, any individual protein and complex can bind to each other as far as the complex does not contain the present unit protein. The following are the assumptions related to the reactions:

1) Any two proteins/complexes can bind to form a new complex as far as these two proteins/complexes do not contain the same unit protein.
2) Initially there exist only N types of unit proteins;
3) Eventually a *final* complex containing one piece of each unit protein is formed as the inhibitor/repressor to the unit proteins.

Model 3: protein/complex containing consecutive unit proteins bind

In model 3, all individual proteins are numbered and only those with consecutive indices can form a link in between to form new complexes. The following are the assumptions related to the reactions:

1) No complex can contain any common unit proteins.
2) Any two proteins/ complexes can bind to form a new complex if the reactants are composted of different proteins and in a consecutive order, e.g. 12 and 34 to form 1234 or 1 and 23 to form 123.
3) Initially there are only N types of unit proteins;
4) Eventually a *final* complex containing one piece of each unit protein is formed as the inhibitor/repressor to the unit proteins.



#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Model 1 & Model 2	1	2	3	4	5	1	1	1	1	2	2	2	3	3	4	1	1	1	1	1	1	2	2	2	3	1	1	1	1	2	1
						2	3	4	5	3	4	5	4	5	5	2	2	2	3	3	4	3	3	4	4	2	2	2	3	3	2
																3	4	5	4	5	5	4	5	5	5	3	3	4	4	4	3
																										4	5	5	5	5	4
																															5
Model 3	1	2	3	4	5	1	2	3	4	1	2	3	1	2	1																
						2	3	4	5	2	3	4	2	3	2																
										3	4	5	3	4	3																
													4	5	4																
															5																

Figure 3: The indices of all species for N=5 in three models: the first row is the collection of all indices, and a vertical bar of five cells shows the included unit proteins for each species.

Figure 3 shows a table of all the intermediate complexes of these three models for an example having 5 initial unit proteins. From the table, we can see that Model 1 and Model 2 share the same types of formation though the binding mechanisms are different. In these two models, the order of the individual proteins does not matter, i.e. all individual proteins play exactly the same role. Model 3 forms less intermediate complexes due to the restrictions.

3. Algorithm and implementation

We will use the Law of Mass Action to generate system of Ordinary Differential Equations (ODEs) according to the specific chemical reactions. Below are the general rules.

1) The LHS of the differential equation is the rate of change for the concentration of all the involved substances including both reactants and products.
2) For a particular substance, the RHS of the differential equation for this substance will have a term for each involved reaction of this substance as $k[A]^a[B]^b$, where k is the rate constant, $[A]$ and $[B]$ are the concentration of the **reactants**, and a and b are coefficients in the chemical equation for the reaction, which is usually 1 for the binding models involved in this article.
3) The sign of k is decided by the production (+) or consumption of the substance (-) in the reaction.

The number of ODEs is determined by the number of species of the biological systems. For each ODE, the right hand side is the summation of all the terms related to the rate of the current species, which is from two types: the *general* binding reaction terms and the *specific* reaction terms such as transcription, translation, transportation, phosphorylation and so on. The first part will be automatically generated with our algorithms and the second part will be manually handled.

In order to automatically generate the term for the general binding reactions, we need to first order and number all existing species, thus we could locate any individual species conveniently for involved reactions.

1. Number and order the species

We need to number and order all the species in the system as $y(1), \ldots, y(N_v)$, where $y(i)$ denotes the concentration of the i^{th} species, $N_v = \sum_{i=1}^{N} C_N^i = 2^N - 1$ is the total number of variables, N is the number of the initial unit proteins, and the notation C_N^i refers to the number of possible combinations of i objects from a set of N objects. Models 1-3 will initially use the same total number of variables and the restriction for Model 3 that the complex must have unit proteins with consecutive indexes will be included by setting reaction rate constants of non-existing complexes zero. We divide the total number of species into N segments and each segment includes the C_N^i species, which all contain exactly i individual proteins for $i = 1, \ldots, N$.

To store species in each segment, we use a recursive subroutine to order all the possible combinations. As an example, for the case with five unit proteins, all the species that are ordered and indexed can be seen in the table in Fig. 3. The code in generating this kind of table is provided in the Appendix 7.1.

2. Find and code the reaction between species

To mathematically describe reactions between any two species, we need to automatically code the general rules for binding of any two species and manually code reactions such as transcription, translation, transportation, inhabitation, phosphorylation, degradation, and so on. To decide if any two species will react with each other, we will use the following strategies. According to the index table generated previously, we allocate two multi-dimensional variables $A(N_v, N_v)$ and $B(2, N_v, N_v)$. Here variable A will store the rate constants and variable B will store the indices of the two reactants. To put value in these two variables, we use two loops looping around all the species from 1 to N_v. For example, when we are at the i^{th} position of the outer loop and the j^{th} position of the inner loop, we need to decide if these two species $y(i)$ and $y(j)$ will react by checking the following rules.

a) If $y(i)$ and $y(j)$ will form a species according to the restriction of the three models, we put the reaction rate into the matrix A at position (i, j) and the indices of $y(i)$, and $y(j)$ into matrix B at positions $(1, i, j)$ and $(2, i, j)$. Note this is for species i.

b) If $y(i)$ contains $y(j)$, i.e. $y(j)$ can form $y(i)$ by binding to the third species $y(k)$, we then put the reaction rate into matrix A at position (i, j) and the indices of $y(j)$ and $y(k)$ into matrix B at positions $(1, i, j)$ and $(2, i, j)$. The pseudo code is listed in the Appendix 7.2. Note the code for generating variables A and B will be only called once, since the resulting variables will be stored in memory. The ODE solver can directly access them at each time step. For instance, for species i, at each time step, we can use the following equation to include all the general reactions involving species i.

$$\frac{dy(i)}{dt} = \cdots + \sum_{j=1}^{N_v} A(i,j) y(B(1,i,j)) y(B(2,i,j)) \tag{4}$$

In a negative feedback loop where the final complex will inhibit the generation of the unit proteins, we use the Hill coefficient n as in Eq. (5) to control the strength of negative feedback. For $i = 1, 2, \ldots, N$, we have

$$\frac{dy(i)}{dt} = \frac{(k_m(i))^n}{(k_m(i))^n + (y(N_v))^n} + \cdots \tag{5}$$

where k_m is the repressing constant for the i^{th} individual protein.

3. Solve the resulting non-linear system of ODEs problems

The system of ODEs of the proposed problems is composed of N_v non-linear equations. For $N_v = O(10^3)$ or less, we can use the ODE solvers in Matlab, e.g. non-stiff ODE solvers such as ode45 (Dormand-Prince), ode23 (Bogacki-Shampine) and ode113 (Adams), or stiff ODE solvers such as ode15s (numerical differentiation formulas), ode23s (Rosenbrock), ode23 (Trapezoidal), and ode23tb (Trapezoidal+backward differentiation formula of order 2). The numerical tests show that these solvers, although vary slightly in speed and time steps, can all provide similar results. We recommend ode23 here, as it stands out as the most efficient one for the problems we solved.

We used two tricks to improve the efficiency of the solver. One is to use global variables instead of function arguments to supply the pre-determined parameters, and the other is to convert the heavily-loaded products in Eq. (4) into a simplified form with only non-zero terms.

However, when the problem size increases, Matlab is not able to solve the proposed problem efficiently, and we need to resort to solvers with better computing efficiency. There are many ODE packages written in C/C++ or Fortran available and the most popular one is CVODE [20], which is included in SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers) package maintained by Laurence Livermore National Lab [21]. The preliminary results tested on a classic Goodwin oscillation problem (no binding but transition) with 500 variables showed improvement in speed in the scale of $O(10)$ compared with solving the same problem with Matlab on the same machine (8-core Intel(R) Xeon(R) CPU E5440 @ 2.83GHz with 2G mem per core).

Another option, which is still under consideration, is to use the matrix Riccati differential equation [22, 23] and its available explicit solution to address the proposed problems. Leipnik [22] provided a convenient explicit solution to a canonical form of the self-adjoint Matrix Riccati differential equation with constant coefficients. Kerner [23] showed the fundamental laws of chemical kinetics for either open or closed systems with an arbitrarily large number of reactants can be represented as a system of Riccati-like differential equations and this system can be reduced to a linear form exactly. We are trying to circumvent two difficulties associated with this application. The first difficulty is how to include *specific* reactions such as transcription, translations, transportation, phosphorylation into the Riccati-like system. And the second is how to numerically evaluate the explicit solution, which involves heavy expenses in evaluating the matrix exponential.

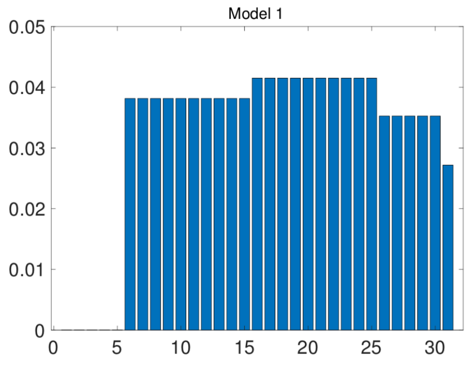
Most mathematical models of cellular biochemical networks consider nonlinear terms such as Michaelis-Menten or Hill-type reactions. Our approach is to explicitly consider all the basic bimolecular reactions that comprise the system. It should be noted that the nonlinear terms used in other models are simplifications of larger systems of the form we consider here. Thus we expect that, for correct choices of parameters, our models should be able to reproduce all the behaviors seen in the simpler models, and additionally, show other types of behavior not seen in these models. Of course, the downside is that our models are more complex, and therefore take more time to simulate and are less likely to be amenable to mathematical analysis. However, we note that, with improvements in computing power, and proper numerical techniques, the simulation of these larger models is still not prohibitively time consuming. Thus, while the large number of variables in our systems seem to be prohibitively large for standard mathematical analysis, the fact that our models restrict nonlinearities of molecular interactions to quadratic terms, may help with mathematical analysis.

4. Results and discussion

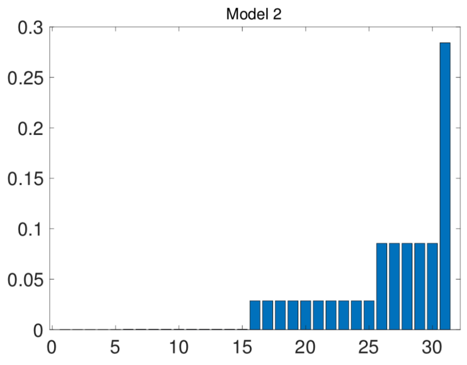
In this section, we consider a system with initially 5 unit proteins with concentration 1M. We investigate the behavior of the system in the scenarios without feedback and degradation and with feedback and degradation. Results using 4 unit proteins with 15 species have been reported in [5], and current results using 5 unit proteins with 31 species will give a different perspective. It will reconfirm the claim we made before and give insights about how an increment of system size will affect the behavior of the system.

1. Distributions and concentration study for models without feedback

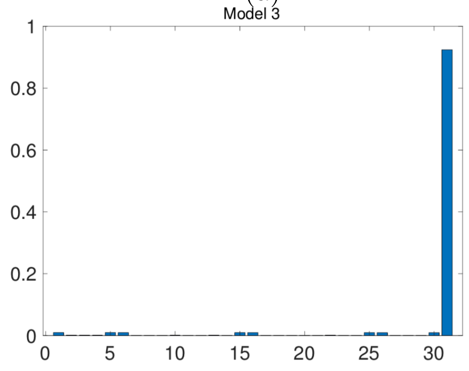
We first studied how binding procedure alone determines the concentration of the final complex. For these three models, we obtained different concentration dynamics for 100s running time as seen in Fig. 4(d). It can be seen only in model 3 the concentration of the final complex approaches 1M, indicating that all the unit proteins eventually participated in forming the final complex. To explain this, we plot the species distribution for all three models at t=100s as illustrated in the bar graph of Fig. 4(a)-(c). The plots explain the observation well, as the distributions of the intermediate complexes of Model 1 and 2 showed that the binding stopped when the remained species in the system cannot bind to each other anymore. This observation is also consistent with the results we reported in [5] using 4 unit proteins.



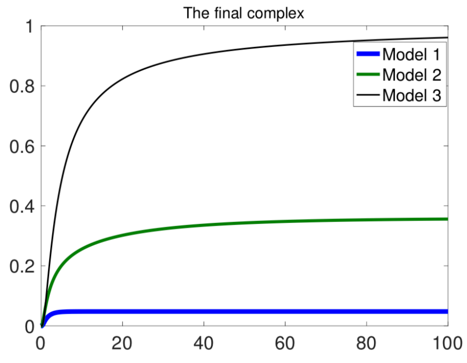
(a)



(b)



(c)



(d)

Figure 4: Distributions of all species at steady status and dynamic of concentrations of the final complex. (a)-(c): distribution (in ratio) for all species in model 1-3 after $t = 100s$ running time; the horizontal axis is the indices of all involved species; (d) the concentration of the final complex in all three models through the whole process ($t = 0 - 100s$).

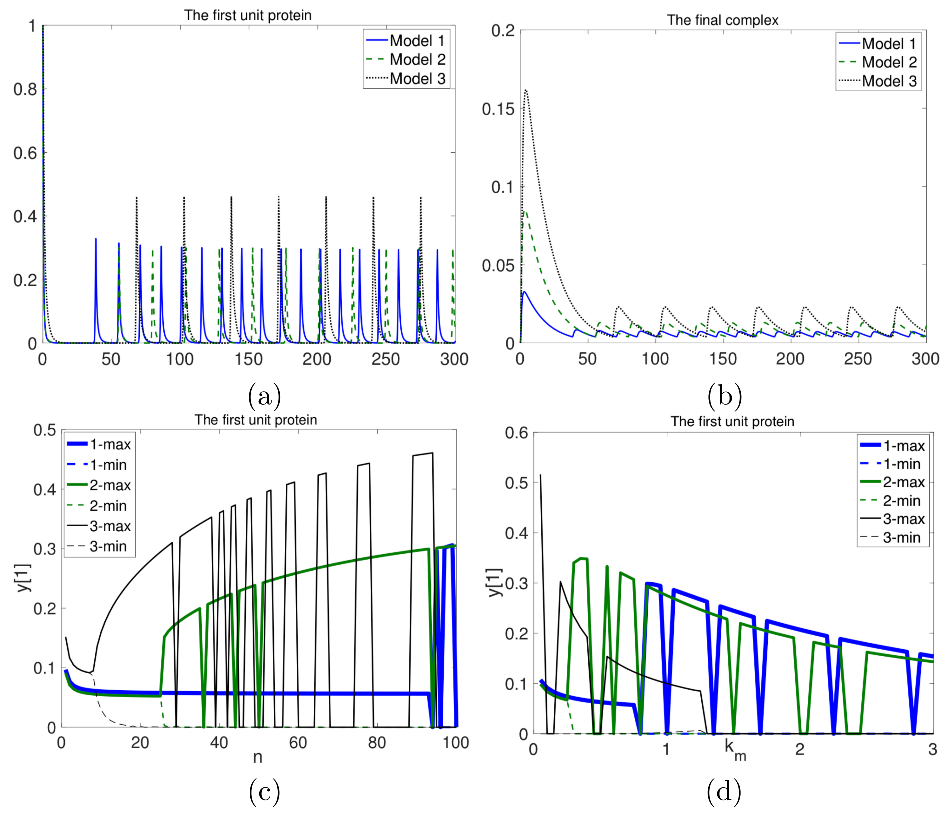


Figure 5: The oscillated concentration and bifurcation plots of models 1-3 (solid line for maximum values and dashed line for minimum values; a different maximum and minimum indicates the occurrence of oscillation). (a) the concentration of the initial unit proteins, (b) the concentration of the final complex, (c) the bifurcation plot of models subject to Hill constant n, and (d) the bifurcation plot of models subject to parameter k_m; used parameters: the degradation constant $k_d = 0.4$ except $k_d(N_v) = 0.05$, $k_m = 0.8$ except it has a range for (d), $n = 94$ except it has a range for (c).

2. Concentration study for models with feedback

Now we include the feedback in our system, the feedback loop is similar to what was illustrated in Fig. 1. This simplified process is proposed first by Goodwin [19] and was analyzed in detail by Hastings et al. [24]. We here include the binding procedure into Goodwin's model. The initial unit proteins will form intermediate complexes and finally the final complex, which will repress the generation of the individual protein (e.g. transcription and translation). The corresponding differential equations are given as:

$$\frac{dy(i)}{dt} = \frac{(k_m(i))^n}{(k_m(i))^n + (y(N_v))^n} - k_i y(i) \qquad \text{for } i = 1, \ldots, N \tag{6}$$

$$\frac{dy(i)}{dt} = g_i(\mathbf{y}) - k_i y(i), \qquad \text{for } i = N+1, \ldots, N_v \tag{7}$$

where $\mathbf{y} = (y(1), y(2), \ldots, y(N_v))$, and g_i is a function involving reactants in all reactions related to $y(i)$.

After varying different parameters like Hill coefficient n, the repressing constant k_m, and degradation rate k_i, we obtained oscillations for all three models. We plot the concentration of the first individual protein for all three models in Fig. 5(a) and the concentration of the final complex in Fig. 5(b) for $t = 0 - 300$s. For all models, Hill coefficient n, combined with some other parameters e.g. the repressing constant k_m and degradation rate k , will determine the oscillation and its frequency and amplitude. To study this, we show the bifurcation plot (maximum value and minimum value of y[1] differ thus indicate an oscillation) in Figs. 5(c), from which we could see Model 1 starts the oscillation when $n \geq 10$, Model 2 starts the oscillation when $n \geq 25$, and Model 3 starts the oscillation when the $n \geq 94$. Figure 5(d) shows that the occurrence of oscillation is also related to the repressing parameter k_m. With the increment of n, these oscillations are more likely to occur but still on and off. The phenomenon observed here can be explained with bar graphs in Fig. 4(a)-(c). In model 3, all individual proteins eventually are used to form final complex; in model 2, the final complex is about 27% of the total existing species at steady status; and in model 1, the final complex is less than 3% of the total. These observations show that the concentration of $y(N_v)$ is best-controlled by the concentration of unit proteins $y(i), i = 1, \ldots, N$ in model 3, followed by model 2 and then model 1. The conclusion is that the smoother the pathway of forming final complex from unit proteins is, the more likely the oscillation will happen. The models using $N = 4$ has been reported in our previous work [5]. A back to back comparison shows that the larger the N, the more difficult it will be for the oscillation to occur.

5. Conclusion

Combinatorial complexity is vital but often ignored in mathematical models of biochemical reaction networks by making simplifying assumptions that may or may not be appropriate. This paper presents an algorithm to conveniently include the combinatorial complexity in the coupled reaction networks and study its influence under various restrictions. By considering the general binding mechanism, this algorithm can automatically generated system of differential equations for the concentration of all the intermediate species determined by combinatorics while leave the flexibility to manually incorporate specific reactions such as translation, transcription, transportation, etc. Numerical simulation is conducted to study the reaction of a system composed initially five unit proteins with three different binding schemes. The results indicate that different binding schemes will significantly change the reaction speed and distribution of species in the system, and these changes will fundamentally affect the negative feedback loop in period, magnitude, and occurrence of the biological oscillations.

6. Acknowledgement

This project is part of the REU summer research conducted by undergraduate student Xin Yang, who was sponsored by Hamilton's scholar program and Undergraduate Research Assistantship (URA) program from Southern Methodist University. The authors thank Professor Daniel Forger from University of Michigan for useful suggestions and discussions.

7. Appendix

The difficulty of the coding is at how to number all the variables and how to locate the corresponding variable when necessary. For example, if we know the index of a corresponding variable, what are the involved unit proteins? Or if we know the complex $y(i)$ composed of n_i unit proteins $y(i_1)$, $y(i_2)$,...,$y(n_i)$, how do we find its index i? The first difficulty is solved by generating a table with dimension $(2^N - 1) \times N$, which matches index of the $2^N - 1$ variables and included unit proteins. With this table in hand, the second difficulty can be solved easily as seen in the subroutine *index*. The following subroutines in Section 7.1 realize these functions.

Another challenge is how to write the right hand side of the system of ODE. Since the RHS will be called repeatedly for any ODE solver, it is worthwhile to save them into two tables. The first table will have the dimension $N \times (2^N)$, saving the related information as the rate of the N unit variables. The second table will have the dimension $(2^{N-1}) \times 2N$, storing the related information as the rate of the 2^{N-1} complexes. The pseudo code and related explanation in Section 7.2 answers this question.

7.1. The pseudo code for generating the index table

This section gives the pseudo code for generating the index table I2A(N_v,N) such as in the top portion Fig. 3 for a given parameter N.

```
% Initialize Global Variables
N=total number of individual proteins;
Nv=total number of species (2^N-1);
I2A(Nv,N)=0;
% The table storing all species and their components

% Local variables used in the loops
% i: number of individual protein;
% j: column index, number of units filled for a species
% indx: the number of completed rows in I2A after ith loop
% indx0: the index of current species
% Array(N): The array storing component of current species

%Assign the first N rows of I2A as individual proteins
I2A(1,:)=[1,0,0,...]
I2A(2,:)=[2,0,0,...]
...
I2A(N,:)=[N,0,0,...]

indx=N
for i=2,...,N
    j=1;
    Array=[1,0,...,0] %starting from 1
    indx0=indx;
    call find_species(i,j,Array,indx0,N,I2A);
    indx=indx+
end

iterative subroutine find_species(i,j,Array,indx0,N,I2A);
% Currently, the table has been filled with indx0 species,
% current species has i individual proteins,
% with j elements filled,
% and the subroutine decides what to do next.

while (# of ele. to fill, (i-j)
       <= # of unit proteins remained (N-Array(j)))
if (# of ele. to fill < # of unit proteins in this species)
    j=j+1;
    Array(j)=Array(j-1)+1;
    Call find_species(i,j,Array,indx0,N,I2A);
elseif (# of ele. to fill = # of unit proteins in this species)
    indx0=indx0+1;
    I2A(indx0,:)=Array; %(one row finished!)
end
Array(j)=Array(j)+1;
end
j=j-1;
```

Explanation of the iterative subroutine
Case 1: Moving forward
Condition:
  1) There are still enough proteins ($N-$Array(j)) left to fill the remained units ($i-j$)
  2) The array is not fully filled ($j < i$);
  Operation:
    1) Move to the next unit; ($j = j + 1$)
    2) Fill the unit with 1 plus the value of last filled number;
      (Array(j)=Array($j-1$) + 1);

Case 2: Reaching the last unit (j=i)
Condition:
  1) $i = j$ and Array(j) $<= N$
  Operation:
    1) Increase the index of current species by 1; (indx0=indx0+1)
    2) Store the current species to the table;
      (I2A(indx0,1:i)=Array(1:i))
    3) Increase the value the last element of Array by one;
      (Array(j)=Array(j) + 1)

Case 3: Exhausting the proteins
Condition:
  1) Not enough individual proteins to fill the current species

```
    (N−Array(j)< i − j);
  Operation:
    1) Go back to the previous column ($j = j - 1$) and exit current subroutine;
    2) Fill in the unit with 1 plus the value of last filled number
       Array(j)=Array(j) + 1;
```

7.2. The pseudo code for generating system of ODEs

The involved number of variables (unit protein and complex), $N_v = 2^N - 1$, and the number of reactions, $N_r = 2^N - N - 1$ since each reaction will create a new variable. We use $y_u(i)$ to represent the concentration of unit protein, and use $y_c(i)$ to represent the concentration of complex. Then we could figure out that the number of reactions, N_u, involving a specific unit protein $y_u(i)$, $i = 1, ..., N$, equals to the number of variables which do not have $y_u(i)$ as its component, calculated as $N_u = \sum_{i=1}^{N-1} C_N^i = 2^{N-1} - 1$. Similarly, the number of reactions, N_c, involving a specific complex $y_c(i)$, $i = 1, ..., 2^N - N - 1$, is N as the summation of the numbers of unit variables included in $y_c(i)$ and not included in $y_c(i)$.

Keeping the above calculation in mind, let $y(i) = y_u(i)$, $i = 1, 2, ..., N$, and $y(i) = y_c(i - N)$, $i = N + 1, ..., 2^N - 1$, taking Model 1 as an example, we could design the following subroutines with the notation and variables.

```
do i=1, $N$
  $\frac{dy(i)}{dt} = \frac{(k_m(i))^n}{(k_m(i))^n - (P^*)^n} - \sum_{j \in C_i} K_{ij} y(i) y(j) - K_d(i) y(i)$
  // $y(i)$ is the concentration of $i$th unit protein
  // $\underline{C_i = \{\text{complexes w/o containing } y(i)\}}$
  // $P^*$ is the final complex as inhibitor
  // $K_d(i)$ is the degrade constant associated with $y(i)$
  // $k_m(i)$ is the repressing constant associated with $y(i)$
  // $K_{ij}$ is the rate constant associated with unit protein $y(i)$ and complex $y(j)$
enddo

do i=$N + 1$,$N_v$
  $\frac{dV(i)}{dt} = \sum_{j \in D_i} K_{jj^*} y(j) y(j^*) - \sum_{k \in E_i} K_{ik} y(k) y(i) - K_d(i) y(i)$
```

```
    // $y(i)$ is the complex
    // $\underline{D_i = \{\texttt{the set of unit proteins contained in } y(i)\}}$
    // $y(j)$ is an unit protein in $D_i$
    // $y(j^*)$ is the complex which forms $y(i)$ with $y(j)$
    // $K_{jj*}$ is the rate constant associated w/ unit protein $y(j)$
and Complex $y(j^*)$
    // $\underline{E_i = \{\texttt{the set of unit proteins w/o contained in } y(i)\}}$
    // $y(k)$ is an unit protein in $E_i$
    // $K_{ik}$ is the rate constant associated w/ unit protein $y(k)$ and
Complex $y(i)$
    // $K_d(i)$ is the degrade constant associated with $y(i)$
enddo
```

References

[1] D. B. Forger and C. S. Peskin, *A detailed predictive model of the mammalian circadian clock*, in: Proceedings of the National Academy of Sciences **100** (2003), no. 25, 14806–14811.

[2] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek, *Bionetgen: software for rule-based modeling of signal transduction based on the interactions of molecular domains*, Bioinformatics **20** (2004), 3289–3291.

[3] J. Schaff, C. C. Fink, B. Slepchenko, J. H. Carson, and L. M. Loew, *A general computational framework for modeling cellular structure and function*, Biophysical Journal (1997), 1135–1146.

[4] L. Lok and R. Brent, *Automatic generation of cellular reaction networks with moleculizer* 1.0, Nature Biotechnology **23** (2005), 131–136.

[5] D. DeWoskin, W. Geng, A. R. Stinchcombe, and D. B. Forger, *It is not the parts, but how they interact that determines the behaviour of circadian rhythms across scales and organisms*, Interface Focus **4** (2014), no. 3.

[6] T. Stankovski, T. Pereira, P. V. E. McClintock, and A. Stefanovska, *Coupling functions: Universal insights into dynamical interaction mechanisms*, Rev. Mod. Phys. **89** (2017), 045001.

[7] D. DeWoskin, J. Myung, M. D. C. Belle, H. D. Piggins, T. Takumi, and D. B. Forger, *Distinct roles for gaba across multiple timescales in mammalian circadian timekeeping*, Proceedings of the National Academy of Sciences (2015).

[8] J. Myung, S. Hong, D. DeWoskin, E. De Schutter, D. B. Forger, and T. Takumi, *Gaba-mediated repulsive coupling between circadian clock neurons in the scn encodes seasonal time*, Proceedings of the National Academy of Sciences **112** (2015), no. 29, E3920–E3929.

[9] J. H. Abel, K. Meeker, D. Granados-Fuentes, P. C. St. John, T. J. Wang, B. B. Bales, F. J. Doyle, E. D. Herzog, and L. R. Petzold, *Functional network inference of the suprachiasmatic nucleus*, Proceedings of the National Academy of Sciences **113** (2016), no. 16, 4512–4517.

[10] J. K. Kim, *Protein sequestration versus hill-type repression in circadian clock models*, IET Systems Biology **10** (2016), 125–135.

[11] M. C. Leung, M. Hutson, A. W. Seifert, R. M. Spencer, and T. B. Knudsen, *Computational modeling and simulation of genital tubercle development*, Reproductive Toxicology **64** (2016), 151–161. 44th Annual Conference of the European Teratology Society.

[12] M. C. Leung, A. C. Procter, J. V. Goldstone, J. Foox, R. DeSalle, C. J. Mattingly, M. E. Siddall, and A. R. Timme-Laragy, *Applying evolutionary genetics to developmental toxicology and risk assessment*, Reproductive Toxicology **69** (2017), 174–186.

[13] G. Wang and C. S. Peskin, *Entrainment of a cellular circadian oscillator by light in the presence of molecular noise*, Phys. Rev. E **97** (2018), 062416.

[14] P. Fletcher, R. Bertram, and J. Tabak, *From global to local: exploring the relationship between parameters and behaviors in models of electrical excitability*, Journal of Computational Neuroscience **40** (2016), 331–345.

[15] S. S and K. Sriram, *Hypothesis driven single cell dual oscillator mathematical model of circadian rhythms*, PLOS ONE **12** (2017), 1–28.

[16] A. R. Stinchcombe and D. B. Forger, *An efficient method for simulation of noisy coupled multi-dimensional oscillators*, Journal of Computational Physics **321** (2016), 932–946.

[17] S.-A. Bae, A. Acevedo, and I. P. Androulakis, *Asymmetry in signal oscillations contributes to efficiency of periodic systems*, Critical Reviews&trade; in Biomedical Engineering **44** (2016), no. 3, 193–211.

[18] J. D. Murray, Mathematical Biolgy I. An Introduction, Springer, third ed., 2001.

[19] B. C. Goodwin, *Oscillatory behavior in enzymatic control processes*, Advances in Enzyme Regulation **3** (1965),425–437.

[20] S. D. Cohen and A. C. Hindmarsh, *Cvode, a stiff/nonstiff ode solver in c*, Comput. Phys. **10** (1996), 138–143.

[21] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, *SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers*, ACM Transactions on Mathematical Software (TOMS) **31** (2005), no. 3, 363–396.

[22] R. B. Leipnik, *A canonical form and solution for the matrix riccati differential equation*, The Journal of the Australian Mathematical Society. Series B. Applied Mathematics **26** (1985), no. 3, 355–361.

[23] E. H. Kerner, *A dynamical approach to chemical kinetics: Mass-action laws as generalized riccati equations*, The Bulletin of Mathematical Biophysics **34** (1972), 243–275.

[24] S. Hastings, J. Tyson, and D. Webster, *Existence of periodic solutions for negative feedback cellular control systems*, Journal of Differential Equations **25** (1977), no. 1, 39–64.

Weihua Geng and Xin Yang:
Department of Mathematics, Southern Methodist University
Dallas, TX 75275, USA
E-mail address: wgeng@smu.edu
E-mail address: xiny@smu.edu