Stochastic Simulation Methods for Solving Systems with Multi-State Species

Zhen Liu

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Application

Yang Cao, Chair
T. M. Murali
Adrian Sandu

May 6, 2009
Blacksburg, Virginia

Keywords: SSA, StochSim, multi-state, hybrid method, rule-based modeling

Stochastic Simulation Methods for Solving Systems with Multi-State Species

Zhen Liu

(ABSTRACT)

Gillespie's stochastic simulation algorithm (SSA) has been a conventional method for stochastic modeling and simulation of biochemical systems. However, its population-based scheme faces the challenge from multi-state situations in many biochemical models. To tackle this problem, Morton-Firth and Bray's stochastic simulator (StochSim) was proposed with a particle-based scheme. The thesis first provides a detailed comparison between these two methods, and then proposes improvements on StochSim and a hybrid method to combine the advantages of the two methods. Analysis and numerical experiment results demonstrate that the hybrid method exhibits extraordinary performance for systems with both the multi-state feature and a high total population.

In order to deal with the combinatorial complexity caused by the multi-state situation, the rules-based modeling was proposed by Hlavacek's group and the particle-based Network-Free Algorithm (NFA) has been used for its simulation. In this thesis, we improve the NFA so that it has both the population-based and particle-based features. We also propose a population-based method for simulation of the rule-based models.

The bacterial chemotaxis model has served as a good biological example involving multi-state species. We implemented different simulation methods on this model. Then we constructed a graphical interface and compared the behaviors of the bacterium under different mechanisms, including simplified mathematical models and chemically reacting networks which are simulated stochastically.

Contents

1 Overview **1**

2 Comparison between SSA and StochSim **6**

2.1 Background 6

2.1.1 SSA 6

2.1.2 StochSim 7

2.2 Accuracy Analysis on StochSim 10

2.3 Efficiency Comparison 15

2.4 Multi-State Situation 17

2.4.1 Multi-State Species 17

2.4.2 The Hybrid SSA 19

2.5 Numerical Experiments 20

2.5.1 Bacteria Chemotaxis Model 20

2.5.2 Efficiency Comparison on a Simple Example 21

3 A Hybrid Strategy Combining Improved StochSim and SSA **26**

3.1 A Hybrid Method 26
3.1.1 An Improved StochSim 27
3.1.2 Combination of SSA and StochSim 30
3.2 Numerical Experiments 32

4 Rule-Based Modeling and Simulation **36**
4.1 Background 36
4.1.1 Rule-Based Modeling 36
4.1.2 Network-Free Algorithm 37
4.2 Population NFA (PNFA) 39
4.3 Full-Scale SSA (FSSSA) 41
4.4 Comparison of Simulation Methods 43
4.5 Numerical Experiments 44

5 Biological Example: Bacterial Chemotaxis **47**
5.1 Background 47
5.2 Biochemical Models 49
5.3 Numerical Experiments on the Complete Model 55
5.4 Graphical Interface 56
5.4.1 Models of Mechanisms 58
5.4.2 Experiments 61

6 Conclusion and Future Work **62**

List of Figures

2.1 Time step comparison between SSA and the simple procedure 12

2.2 Trajectory comparison between the SSA and StochSim 22

2.3 Histogram comparison between the SSA and StochSim 23

3.1 Flowchart of the hybrid method combining the SSA and StochSim 33

4.1 Histogram comparison between the PNFA and FSSSA 46

5.1 Bacterial chemotaxis mechanism 48

5.2 Chemical reacting network of bacterial chemotaxis 50

5.3 Screen print of graphical interface 57

5.4 Response curve for bacterial chemotaxis 60

List of Tables

2.1 Numerical results on chemotaxis model by the SSA and StochSim 21

2.2 CPU time comparison among the SSA, StochSim and HSSA 24

3.1 Numerical result comparison 1 among the hybrid method, SSA and StochSim 34

3.2 Numerical result comparison 2 among the hybrid method, SSA and StochSim 34

3.3 Numerical result comparison 3 among the hybrid method, SSA and StochSim 35

4.1 Comparison of all methods 43

4.2 Population distribution on different states 45

5.1 Reaction list of the simplified chemotaxis model 51

5.2 Translated reaction list of the simplified chemotaxis model for the SSA . . . 52

5.3 Reaction list of the complete chemotaxis model 54

5.4 Replacement rules for the complete chemotaxis model 55

5.5 CPU time comparison on the complete chemotaxis model 56

5.6 Numbers of steps that four model bacteria take to arrive at the top area . . 61

Chapter 1

Overview

In many biochemical systems where the numbers of molecules and reactions involved are huge and the dynamics of the system are difficult to analyze theoretically, simulation techniques are very important. Conventionally, models of biochemical systems are formulated using reaction rate equations (RREs), and deterministic and continuous methods are used for the simulations. However, in microscopic systems species with small copy numbers can result in dynamical behavior that is discrete and stochastic. Therefore, in recent years more and more attention has been paid to the stochastic and discrete modeling and simulation methods [1, 2, 3, 4].

Gillespie's Stochastic Simulation Algorithm (SSA) [5, 6] is one of the most famous algorithms for chemically reacting systems. It is an exact simulation scheme as it follows the same probability distribution as ruled by the chemical master equation. The state variables in the SSA usually represent species populations. Thus it is also a *population-based* method. This method was successfully applied in McAdams and Arkin's work [1, 2], and since then has been widely used to simulate the kinetics of different biochemical systems. Later progress [7, 8, 9] has been made to improve the implementation of the SSA. Approximation methods, particularly the tau-leaping methods [10], have been proposed to improve efficiency at a little expense of accuracy.

However, as mathematical models of biological systems become more complicated and realistic, traditional SSA begins to exhibit low efficiency in many cases. One of the most challenging cases is called *combinatorial complexity* in the field of computational biology [11, 12]. In cellular regulatory systems, due to the multi-component composition of biologically reacting agents, proteins can bind and modify each other in many ways [11]. This feature is also known as the *multi-state situation* of proteins and often causes great computational cost for the SSA. One example of the multi-state situation is in the modeling of bacterial chemotaxis, one of the most interesting and well-studied biological phenomena [13]. In the chemical regulatory pathway of bacterial chemotaxis, the enzyme protein $TTWWAA$, which plays a major role, can bind to various kinds of other small proteins and undergo different levels of (de)phosphorylation and (de)methylation reactions. Different bindings lead to different biochemical properties and should be represented by different states. The combination number of different bindings leads to a great number of states in which the enzyme can possibly be. For example, an enzyme with 10 binding sites may have a total of $2^{10} = 1024$ states. As in the SSA each state is assigned one species, the number of both species and reactions can become very large. Since the claimed lowest time complexity of the SSA in a single step is $O(\log(\mathcal{M}))$ where $\mathcal{M}$ is the number of reactions [7, 8, 14, 15, 16], simulation with the SSA may be very inefficient.

The multi-state situation presents a great challenge to the modeling and simulation of biochemical systems. In order to answer this challenge, *particle-based* modeling and simulation methods have been proposed [17, 18]. One of the earlier stochastic simulators of this kind is StochSim, which was developed by Morton-Firth and Bray [19, 20, 21], and successfully applied in the stochastic modeling and simulation of bacterial chemotaxis. An important feature of StochSim is its particle-based nature. It treats all reacting molecules as individual objects with their own properties, such as conformation states, velocity, spatial information and so on. Such a special feature makes StochSim extendable for handling multi-state variables and spatially inhomogeneous stochastic simulation, and its efficiency is not affected by the number of reactions present in the system [22]. However, StochSim has exhibited low

efficiency in systems containing large number of molecules [23, 24].

Then very naturally we have an important question: Are the SSA and StochSim equivalent? This question was first discussed in [21], which showed an equivalence of the physical assumptions of the two methods. However, even though both methods are based on equivalent fundamental physics assumptions, mathematically they may still be different, particularly in their computational efficiency. Later, another comparison [24] was published to show efficiency differences between them. That comparison was mostly based on a particular model. Theoretical comparison was still needed to understand their differences in accuracy and efficiency. In this thesis, we will present a detailed comparison based on fundamental probability analysis [23]. Our analysis reveals that if the time step in StochSim is selected very small, the probability used in StochSim is a first-order approximation to the corresponding one in the SSA. In other words, they are not equivalent but can be very close. According to the analysis, the SSA is more efficient in general, especially for systems with a high total population, because StochSim usually uses a small and fixed step size. However, when multi-state variables are involved in a system, StochSim can be much more efficient. We explain this from complexity analysis and numerical experiments. Also, as an initial attempt to develop efficient algorithms that can combine the advantages of them, we proposed the Hybrid SSA (HSSA) for cases where the populations of the multi-state species are low. Note that although it is a hybrid strategy, it is still a version of SSA and does not incorporate any part of procedure from StochSim.

Based on the comparison between the SSA and StochSim, we can see that each algorithm has its own advantages in simulating different types of models. Nevertheless, with the fast development of computational biology, biochemical models with certain features are found not suitable for either algorithm, but require an algorithm that possess advantages from both sides. To be specific, a model system may contain several multi-state species, and at the same time has a large number of molecule populations. We further developed a new hybrid method combining the SSA and StochSim according to their characteristics. This method works by dividing a system into two subsystems: One is simulated by the SSA and the other

is simulated by StochSim. Numerical experiments showed that the hybrid method exhibits higher efficiency than both the SSA and StochSim in most cases.

Another way to deal with multi-state situation is called *rule-based modeling* [25]. This modeling strategy was proposed by William Hlavacek's group at Los Alamos National Laboratory, and the BioNetGen was developed as one rule-based modeling platform [26]. The original BioNetGen provides software that allows users to conveniently build rule-based models and then simulate the models with ordinary differential equations or the SSA. However, for stochastic simulations with the SSA, often the reaction network generated by the rule-based model can become too large. Since the rule-based models are similar to models represented and simulated by StochSim, a particle-based simulation method such as StochSim rather than the SSA would be more efficient. However, StochSim usually takes very small step sizes and is prone to a great number of null events in simulation. Danos et. al. [27] developed a method specific for rule-based models. It is modified from the original SSA by substituting reactions with rules and using a particle-based simulation scheme, so it no longer suffers from null events. Later, Yang et. al. [28] generalized this method by enabling it to deal with adjustable reacting rates of rules. A rule can have different rate constants for different states of reacting species[28]. Following their work, Sneddon et. al. developed a new rule-based model simulator called Network-Free Stochastic Simulator (NFsim) [29], which utilizes the methods presented in [27, 28]. This method is called the Network-Free Algorithm (NFA).

In our experiments, the Network-Free method was first improved and implemented with sample models and the bacterial chemotaxis, and exhibited low efficiency. Through brief analysis on the method, it is discovered that a large portion of CPU time is spent on updating and maintaining the data structures in the implementation. To further explore for the best simulation method for the rule-based models, we developed the Full-Scaled SSA (FSSSA) by using population-based scheme to simulate rule-based models including the bacterial chemotaxis model. While relevant experiments and analysis is still undergoing, the current results are not sufficient to provide conclusion yet.

Simulations on the rule-based chemical model of the bacterial chemotaxis with both StochSim and FSSSA have been incorporated into a computer-based graphical interface written in C++. For the purpose of comparison, two other mathematical models are also incorporated into the interface. The behaviors of the bacteria simulated by all the models are analyzed.

The thesis is organized as following. In Chapter 2, detailed comparison between StochSim is provided and the Hybrid SSA is proposed. In Chapter 3, a hybrid method combining the improved StochSim and the SSA is proposed. In Chapter 4, two rule-based simulation methods, the population-based NFA and the Full-Scale SSA, are proposed. In chapter 5, the bacterial chemotaxis is studied as a biological example.

Chapter 2

Comparison between SSA and StochSim

This chapter first introduces the SSA and StochSim, and then gives a detailed comparison between the two methods in their accuracy and efficiency. Since it has been shown that StochSim exhibits better efficiency than the SSA with multi-state situations [24], we propose the Hybrid SSA to improve the performance of the original SSA in certain multi-state situations. Last, numerical experiments are done with the bacterial chemotaxis model and another sample model.

2.1 Background

2.1.1 SSA

Suppose the system involves $\mathcal{N}_\mathcal{S}$ molecule species $\{S_1, \ldots, S_{\mathcal{N}_\mathcal{S}}\}$. The state vector is denoted by $X(t) = (X_1(t), \ldots, X_{\mathcal{N}_\mathcal{S}}(t))$, where $X_i(t)$ is the number of molecules of species S_i at time t. $\mathcal{M}$ reaction channels $\{R_1, \ldots, R_\mathcal{M}\}$ are involved in the system. Assume that the system is

well-stirred and in thermal equilibrium. The dynamics of reaction channel R_j is characterized by the *propensity function* a_j and by the *state change vector* $\nu_j = (\nu_{1j}, \ldots, \nu_{\mathcal{N}_S,j})$: $a_j(x)dt$ gives the probability that one R_j reaction will occur in the next infinitesimal time interval $[t, t+dt)$, and ν_{ij} gives the change in the S_i molecule population induced by one R_j reaction.

The dynamics of the system can be simulated by the SSA method[5, 6]. With $X(t) = x$, let $a_0(x) = \sum_{j=1}^{\mathcal{M}} a_j(x)$. On each step, SSA generates two random numbers r_1 and r_2 in $U(0,1)$, the uniform distribution on the interval $(0,1)$. The time for the next reaction to occur is given by $t + \tau$, where τ is given by

$$\tau = \frac{1}{a_0(x)} \log(\frac{1}{r_1}). \tag{2.1}$$

The index j for the next reaction is given by the smallest integer satisfying

$$\sum_{l=1}^{j} a_l(x) > r_2 a_0(x). \tag{2.2}$$

The system states are updated by $X(t+\tau) = x + \nu_j$. The simulation proceeds to the next occurring time, until it reaches the final time. For a system with a large value of $\mathcal{M}$, we can assume that the stoichiometric matrix is sparse, which is usually true for a large chemical reacting system. Then with the best simulation strategy, the time complexity in a single step can be estimated by $O(\log(\mathcal{M}))$ [7, 8, 14, 15, 16].

2.1.2 StochSim

Initialization

As a particle-based algorithm, before simulation StochSim creates objects for all molecules with their own properties according to initial conditions, along with a computed number of *pseudo-molecule objects*. A look-up table is then constructed to describe the reaction possibilities for all reaction channels. The rows of the table list the first reactant and the columns list the second reactant (if the second reactant is a pseudo-molecule, it represents

a uni-molecule reaction). The corresponding entry in the table shows the probability for the two molecules to have a reaction. If there are multiple reaction channels between the two molecules, this entry saves the sum of the probabilities of all involved reaction channels. Note that reactions described in StochSim are actually rules which may include multiple SSA reactions. For the definition of rule, refer to Section 4.1.1.

In order to distinguish uni-molecule reactions from bi-molecule reactions, pseudo-molecules are introduced with a fixed population N_0, which is calculated to make the maximum possibility of uni-molecule reactions equal to that of the bi-molecule reactions. Let k_{1max} be the maximum reaction rate of all uni-molecule reactions, and k_{2max} be the maximum reaction rate of all bi-molecule reactions. Therefore,

$$N_0 = Round(2N_AV \times \frac{k_{1max}}{k_{2max}}), \tag{2.3}$$

where $Round(x)$ represents the positive integer nearest to x, N_A is the Avogadro constant, and V is the volume of the system.

Then, probabilities in the look-up table are calculated with the following formula. For a uni-molecule reaction,

$$p_{1j} = \frac{k_{1j}N(N + N_0 - 1)\Delta t}{N_0}, \tag{2.4}$$

and for a bi-molecule reaction,

$$p_{2j} = \frac{k_{2j}N(N + N_0 - 1)\Delta t}{2N_AV}, \tag{2.5}$$

where N is the total number of all real molecules in the system, k_{1j} and k_{2j} are respectively the uni-molecule rate constant and the bi-molecule rate constant for reaction j, and Δt is the time step size which is pre-determined from user's input.

In the last step before simulation, the time step size Δt is optimized. The criteria is that the maximum probability in the look-up table must be smaller than a constant $MAXP$. In the real implementation of StochSim [30], $MAXP$ is set as 0.599. and the corresponding formula for Δt is given [30] by

$$\Delta t = \frac{MAXP}{k_{1max}N + \frac{k_{2max}N(N-1)}{2N_AV}}. \tag{2.6}$$

This optimized Δt then replaces the previous value, and is used to recalculate all the probabilities in the look-up table.

Note that the total population of all real molecules is also fixed and limits the application of the algorithm. For example, in a system with a binding reaction

$$A + B \longrightarrow C, \tag{2.7}$$

the total number of molecules N will change after firing this reaction. In the implementation [30] of StochSim, N is chosen as the maximum possible total number of molecules, equal to or larger than the actual total population, and certain molecules of dummy species are introduced to compensate the change of total number of molecules. So the reaction (2.7) can be read as

$$A + B \longrightarrow C + Dummy. \tag{2.8}$$

With this simplification, the algorithm keeps N a fixed number.

Simulation

The whole simulation time interval is evenly divided into a series of discrete time steps of a fixed size Δt. In each time step, StochSim proceeds with the following steps.

1. Randomly select the first reactant from all real molecule objects using uniform distribution.

2. Randomly select the second reactant from all real molecule objects and pseudo-molecule objects using uniform distribution.

3. Search the look-up table for possible reactions between the two selected objects. If no corresponding entry is found, StochSim concludes that no reaction occurs in this time step. Otherwise a uniform random number in $(0, 1)$ is generated and compared with the probability retrieved from the table. If the random number is larger, StochSim

concludes that no reaction occurs in this time step. Otherwise there is a reaction between these two molecules. If there is only one possible reaction channel between these two molecules, the reaction is selected to fire. Otherwise, the code selects one of the possible reaction channels in a similar way as the equation (2.2) in the SSA.

4. Update the system accordingly and proceed the simulation to the next time step.

2.2 Accuracy Analysis on StochSim

The implementation details of the SSA and StochSim are quite different. However, it was stated in Shimizu and Bray [21] that these two methods are based on equivalent fundamental physics assumptions. Pettigrew and Resat [24] also showed that these two methods generated similar distribution plots for a particular model. In our work, we analyze StochSim from a different angle. Since the SSA is an exact stochastic simulation scheme, it is natural to first examine the accuracy of StochSim.

Instead of directly comparing it with the SSA, we first compare the SSA with a simple simulation procedure. Suppose that we equally divide the time interval into many small time slices Δt. Because the probability that the R_j reaction channel will fire in the next infinitesimal time interval $[t, t + dt)$ is given by $a_j(x)dt$, when Δt is sufficiently small the probability that one R_j reaction will occur in the time interval $[t, t+\Delta t)$ can be *approximated* by $a_j(x)\Delta t$. We can then implement the following simulation procedure.

Simulation Procedure 2.2.1 *Suppose at time t the system is at $X(t) = x$ and the probability that one R_j reaction will occur in the time interval $[t, t + \Delta t)$ is given by $a_j(x)\Delta t$. In each Δt time slice, a random number r is generated uniformly on interval $[0,\ 1)$ and compared with $a_0(x)\Delta t$. If $a_0(x)\Delta t > r$, one reaction will fire in this small time slice Δt and the reaction channel index j can be selected the same as (2.2) in the standard SSA procedure. We let the time proceed to $t + \Delta t$ and update the state variable by $X(t + \Delta t) = x + \nu_j$.*

Otherwise, no reaction will fire in this time interval. We just let the time proceed to $t + \Delta t$ *with no state variable update.*

One can easily see that this simulation procedure is not exact. If Δt is chosen larger than $\frac{1}{a_0(x)}$, the *probability* $a_0(x)\Delta t$ will be greater than 1, and that is not allowed. Will this procedure be exact if $\Delta t < \frac{1}{a_0(x)}$? No. Note that the simulation procedure 2.2.1 implies that $1 - a_0(x)\Delta t$ is the probability that no reaction will occur in the next time interval Δt. However, one can easily derive that the probability that no reaction will fire in the next Δt for any $\Delta t > 0$ is

$$\mathrm{e}^{-a_0(x)\Delta t} = 1 - a_0(x)\Delta t + \frac{1}{2}\left(a_0(x)\Delta t\right)^2 + O((\Delta t)^3). \tag{2.9}$$

We can see that $1 - a_0(x)\Delta t$ is the first-order approximation of (2.9) and the leading term for the error is given by $\frac{1}{2}\left[a_0(x)\Delta t\right]^2$. The leading term shows that if $a_0(x)\Delta t \leq 0.1$, which gives

$$\Delta t \leq \frac{0.1}{a_0(x)}, \tag{2.10}$$

the error of the first-order approximation can be estimated by 0.005. This Δt may be small enough so that the first-order approximation is acceptable and the histogram generated from the simulation procedure 2.2.1 will be close to that generated by the SSA. However, if Δt has to be one magnitude smaller than $\frac{1}{a_0(x)}$, the chance that no reaction will fire in the next time step $[t,\ t + \Delta t)$ is relatively high. Thus before one reaction really fires in the simulation procedure 2.2.1, there will be several (around 10) steps that no reaction fires at all. This situation can be illustrated in Figure 2.1.

Next we compare the simulation procedure 2.2.1 and StochSim. Let the fixed time steps in StochSim and the procedure 2.2.1 both be Δt. We have the following theorem.

Theorem 2.2.1 *StochSim uses the possibility of* $a_j(x)\delta t$ *to describe one* R_j *reaction to occur in the time interval* $[t, t + \delta t$. *This possibility is as same as the approximated possibility in the simulation procedure 2.2.1, when* δt *is sufficiently small.*

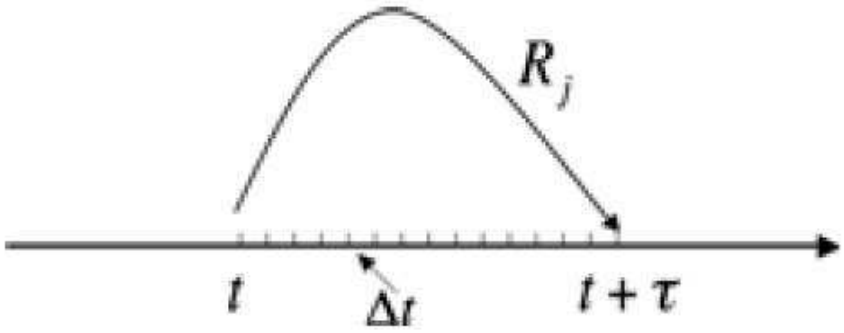

Figure 2.1: The time step comparison between the SSA and the simulation procedure 2.2.1. To ensure the accuracy, for each reaction corresponding to one step τ in the SSA, there must be several (around 10) steps that in the time interval Δt there is no reaction firing in the simulation procedure 2.2.1.

PROOF. In StochSim, two objects are randomly selected in the first two steps. The probability that these two could have a reaction is given by (2.4) or (2.5) if Δt is small.

Let us first look at a mono-molecule reaction R_j with a reactant S_i. In the assumption of simulation procedure 2.2.1, the probability that one R_j reaction will occur in the next time interval $[t, t+\Delta t)$ is given by $a_j(x)\Delta t$, where $a_j(x) = c_j x_i$. On the other hand, in StochSim process R_j is selected when a S_i object is selected in the first step and a pseudo molecular object is selected in the second step. The probability that one S_i molecule is selected in the first step is $\frac{x_i}{N}$. The probability that one pseudo molecule is selected in the second step is $\frac{N_0}{N+N_0-1}$. Multiply them with the equation (2.4). The probability that an R_j reaction will fire in the next time interval $[t,\ t+\Delta t)$ in StochSim is thus given by

$$\frac{x_i}{N} \cdot \frac{N_0}{N+N_0-1} \cdot \frac{k_{1j}N(N+N_0-1)\Delta t}{N_0} = k_{1j}x_i\Delta t. \tag{2.11}$$

Note that $c_j = k_{1j}$ for a mono-molecule reaction [5]. Thus StochSim and the procedure 2.2.1 follow the same probability for a mono-molecule reaction.

For a bi-molecule reaction R_j between reactants S_i and S_k, the procedure 2.2.1 assumes that the probability that one R_j reaction will fire in the time interval $[t, t+\Delta t)$ is given by

$a_j(x)\Delta t$, where $a_j(x) = c_j x_i x_k$. In StochSim, R_j may fire only if S_i and S_k are selected in the first two steps. The probability that one S_i molecule is selected in the first step is $\frac{x_i}{N}$, while the probability that one S_k molecule is selected in the second step is $\frac{x_k}{N+N_0-1}$. The probability that S_k is selected in the first step and S_i is selected in the second step is the same. Thus the probability that one S_i molecule and one S_k molecule are selected in the first two steps is $\frac{2x_i x_k}{N(N+N_0-1)}$. Multiply it with the equation (2.5). The probability that an R_j reaction will fire in the next time interval Δt in StochSim is given by

$$\frac{2x_i x_k}{N(N+N_0-1)} \cdot \frac{k_{2j} N(N+N_0-1)\Delta t}{2N_A V} = \frac{k_{2j}}{N_A V} x_i x_k \Delta t. \tag{2.12}$$

Note that $c_j = \frac{k_{2j}}{N_A V}$ for a bi-molecule reaction between two different species [5]. Again we see that StochSim and the procedure 2.2.1 follow the same probability.

For a bi-molecule reaction R_j between two S_i molecules, the procedure 2.2.1 assumes that the probability that one R_j reaction will fire in the time interval $[t, t+\Delta t)$ is given by $a_j(x)\Delta t$, where $a_j(x) = \frac{1}{2} c_j x_i (x_i - 1)$. In StochSim, R_j may fire only if S_i is selected in both steps. The probability that one S_i molecule is selected in the first step is $\frac{x_i}{N}$, while the probability that a different S_i molecule is selected in the second step is $\frac{x_i - 1}{N+N_0-1}$. Similar to the case of bi-molecule reaction between two different species, these two molecules can be selected with different order. Thus the probability that two S_i molecules are selected in the first two steps is $\frac{2x_i(x_i-1)}{N(N+N_0-1)}$. Multiply it with the equation (2.5). The probability that an R_j reaction will fire in the next time interval Δt in StochSim is given by

$$\frac{2x_i(x_i-1)}{N(N+N_0-1)} \cdot \frac{k_{2j} N(N+N_0-1)\Delta t}{2N_A V} = \frac{k_{2j}}{N_A V} x_i(x_i-1)\Delta t. \tag{2.13}$$

Note that $c_j = \frac{2k_{2j}}{N_A V}$ [1] for a bi-molecule reaction between the same species [5]. Again we see that StochSim and the procedure 2.2.1 follow the same probability. Thus we have the theorem. □

From this theorem we can see that when Δt is sufficiently small, StochSim follows the same

[1] The relation between c_j and k_{2j} in this case is different from the case of bi-molecule reaction between two different species, as pointed out in [5].

probabilities as in the procedure 2.2.1, so it can also be viewed as a first-order approximation to the exact SSA.

We can see this point through analysis on the following example.

Example 1: Suppose a simple system has only one species A and one decay reaction

$$A \xrightarrow{k} X, \tag{2.14}$$

where the population of A remains constant, and k is the reaction rate constant. We let the system proceed by duration of T and count the number of the firings of the only reaction (2.14), denoted by n_f. One can easily see that the possibility for the reaction to fire in the next infinitesimal time $[t, t+dt)$ is given by $|A|kdt$ where $|A|$ is the population of species A, and that the whole process is a Poisson process. Obviously, the theoretical mean value and variance of n_f are both $|A|kT$. Then we use StochSim to simulate this system. Since this system only has a uni-molecule reaction, N_0 is set to ∞ [30]. Therefore according to (2.6), Δt is given by

$$\Delta t = \frac{MAXP}{k|A|}, \tag{2.15}$$

and according to (2.4) and (2.15), the possibility in the look-up table is given by

$$p_1 = k|A|\Delta t = MAXP. \tag{2.16}$$

Note that StochSim evenly divides T into several time steps and in all steps the probability for the reaction to fire is the same, so n_f follows a binomial distribution $B(n, p)$, where

$$n = \frac{T}{\Delta t} = \frac{Tk|A|}{MAXP} \tag{2.17}$$

and

$$p = p_1 = MAXP. \tag{2.18}$$

Therefore with simulation of StochSim, the mean value and variance of n_f are respectively $np = |A|kT$ and $np(1-p) = |A|kT(1 - MAXP)$. Although StochSim generates the correct mean value, the variance is dependent on $MAXP$. Only when $MAXP$ is very small compared

to 1.0, such as 0.1, so that Δt at least satisfies (2.10), the distribution of n_f can be approximately accurate. However, StochSim's real implementation value 0.599 for *MAXP* will apparently make the simulation on this model deviate from the right distribution. Again, in order to obtain the accurate distribution, StochSim has to set Δt sufficiently small.

2.3 Efficiency Comparison

Base on the previous analysis, we are enabled to compare the efficiency of the SSA and StochSim. The computational cost of a dynamic simulation algorithm is composed of two parts: the average computational cost for each time step and the total number of time steps in a simulation. The computational costs for the SSA and the simulation procedure 2.2.1 in each step are both $O(log(\mathcal{M}))$. But there are much more steps in the procedure 2.2.1. Thus we can conclude that the simulation procedure 2.2.1 is an approximation to the SSA with higher computational cost. Obviously it is not a good strategy in stochastic simulation.

Now between the procedure 2.2.1 and StochSim, since their computational costs are proportional to $\frac{1}{\Delta t}$, it is preferred to have a large Δt. However, both are accurate only when Δt is selected sufficiently small. This is a shared restriction on Δt enforced on both procedures. Moreover, StochSim procedure requires all p_{1j}'s in (2.4) and p_{2j}'s in (2.5) be not larger than *MAXP*, a constant not larger than 1.0. This extra restriction on Δt can be even tighter than the first one.

To see this point, let us consider the following example.

Example 2: Suppose there are three species S_1, S_2 and S_3 but only one reaction channel R_1 between S_1 and S_2.

$$S_1 + S_2 \xrightarrow{c} S_3 + S_2. \tag{2.19}$$

Let $X_1 = M_0$, $X_2 = 1$ and $X_3 = 0$, where $M_0 >> 1$. Let $c = 1$. Thus the propensity function for R_1 is $a_1(x) = x_1 x_2$. According to the SSA, the mean time for the next R_1 reaction to

fire is given by

$$\tau_{SSA} = \frac{1}{a_1(x)} = \frac{1}{M_0}. \tag{2.20}$$

In the simulation procedure 2.2.1, the corresponding simulation time step Δt can be given by

$$\Delta t < \frac{0.1}{a_1(x)} = \frac{1}{10M_0}. \tag{2.21}$$

For StochSim, with the maximal value $MAXP = 1.0$ and (2.5), we have

$$p_2 = \frac{k_2 N(N + N_0)\Delta t}{2N_A V} < 1.$$

Here we already know that $c = \frac{k_2}{N_A V} = 1$. Thus StochSim should satisfy the restriction

$$\frac{N(N + N_0)\Delta t}{2} < 1, \tag{2.22}$$

which gives

$$\Delta t < \frac{2}{N(N + N_0)}. \tag{2.23}$$

Here N is the total number of molecules. $N = M_0 + 1$. N_0 is the total number of pseudo-molecules. Since there is no mono-molecule reaction, $N_0 = 0$. Then $\Delta t < \frac{2}{(M_0+1)^2}$. When M_0 is large, this restriction on Δt is much tighter than (2.21). This example demonstrates a situation where StochSim selects a much smaller Δt than the one in the procedure 2.2.1 although Theorem 2.2.1 states that they both follow the same probability when Δt is sufficiently small.

Let $M_0 = 10^4$, equations (2.20) and (2.23) imply that for this example the number of steps StochSim needs is about $5,000$ times as what the SSA needs. The efficiency is very low. Of course this example is an extreme case. According to (2.6), a too small step size in StochSim often arises in situations where some reaction rates related to the species with large populations are relatively small, or there are reactant species with relatively small populations compared to the total populations N. In these cases, the time step in StochSim will be much smaller than the mean time step in the procedure 2.2.1 and the SSA, and cause very low efficiency.

According to the above analysis, we can conclude that StochSim can usually take a much larger number of time steps than the procedure 2.2.1, which is still about one magnitude less efficient than the SSA. On the other hand, we should also note that, the computational cost in a single step for StochSim is $O(1)$ while for the SSA it is at least $O(log(\mathcal{M}))$ [7, 8, 14, 15, 16]. If the average step sizes of the SSA and StochSim are very close to each other and $\mathcal{M} \gg \infty$, StochSim can exhibit higher efficiency. However, the efficiency difference in a single step is usually dominated by the difference in step sizes, especially for multi-scale cases such as Example 2. In this case, the efficiency of StochSim can be much lower than that of the SSA.

2.4 Multi-State Situation

2.4.1 Multi-State Species

Although for a multi-scale system where there can be magnitude difference between populations or reaction rates, the efficiency of StochSim is much lower than that of the SSA, StochSim could have advantages for some systems if its computational cost in a single step is much lower than that of the SSA. A typical situation is when multi-state species are involved. Multi-state species often appear in biological systems where one molecule may change its characteristics depending on changes on its many binding sites, such as phosphorylation or methylation. If a molecule has 10 binding sites, depending on the states of all these binding sites, one molecule may exhibit $2^{10} = 1024$ different states. When it has 20 binding sites, the number of possible states rises to a million. If multi-state species can combine in many different ways, the combinatorial complexity may lead to a very large system size [25, 28, 27]. In the structure of the traditional SSA simulation, each possible state should be assigned with an independent state variable. If each of them may react with other species in the system, $\mathcal{N}_\mathcal{S}$, the number of species, and $\mathcal{M}$, the number of reaction channels, may become dramatically large. Note that the lowest computational cost of the SSA in a single step is $O(\log(\mathcal{M}))$ [7, 8, 14, 15, 16]. If a multi-state species is involved with many reaction

channels, we have $\mathcal{M} = O(N_M)$, where N_M denotes the number of possible states for the multi-state species. N_M can be very large, which results in a large M. The computational cost of the SSA could be very high in this case. However, since StochSim treats molecules as individual objects, it need not introduce a large number of species and reaction channels. Its computational cost for a single step does not change with N_M. Thus when N_M is large, the computational cost in each step of StochSim can be much smaller than that of the SSA, which may compensate the extra cost in the total number of steps as analyzed in the previous section. In that case, StochSim shows advantages over the SSA.

Example 3: Consider a system with three types of species X, Y and E, where E is an enzyme with $1,000$ different states (10 binding sites). Each state has different characteristics for its enzyme ability. And the population of E is small. Three types of reactions are considered.

$$X \xrightarrow{E_n} Y, \tag{2.24}$$

$$E_n \longrightarrow E_{n+1}, \tag{2.25}$$

$$E_{n+1} \longrightarrow E_n, \tag{2.26}$$

where E_n represents E in state n. For the SSA to simulate this system, because each state of E is formulated as an individual species and each of the three reaction types (2.24-2.26) will be correspondingly extended to around 1,000 reaction channels, we have $\mathcal{N}_\mathcal{S} = 1,002$ and $\mathcal{M} \approx 3,000$. However, StochSim can group all these reaction channels into just three channels (2.24-2.26) by treating the 1,000 reaction channels as one. When an E molecule is picked in the first two steps of StochSim, it must have been in a particular state. When one reaction fires and this molecule changes its state, StochSim procedure only needs to change the corresponding state for this molecule.

2.4.2 The Hybrid SSA

For many biological systems, the populations of species are of multi-scale. Most species are present with large or moderate populations and do not have the multi-state problem as discussed in Section 2.4.1. There are a few species with multi-state characteristics. If the multi-state species present with small populations, it is more efficient to treat each multi-state molecule as an independent object. We call this strategy the hybrid SSA (HSSA). The simulation procedure of the HSSA is very similar to that of the standard SSA. The difference lies in the classification of species and reaction channels. A system contains two types of species: normal species and multi-state species. In the HSSA, normal species remain the same as in the standard SSA while each molecule of the multi-state species is stored as an indexed object. Then this system contains three types of indexed reactions.

1. Reactions among normal species. This type remains the same as in the standard SSA.

2. Reactions involved with only one multi-state object. They include uni-molecule reactions of a multi-state object and bi-molecule reactions between a multi-state object and a non-multi-state species.

3. Reactions between two multi-state objects.

In the simulation, the total propensity function $a_0(x)$ is the sum of all reactions. The time step τ and the reaction index j are calculated using (2.1) and (2.2). If the index j points to the first type, the system is updated as in the standard SSA. Otherwise, the firing of a reaction will cause a state change of one or two multi-state objects. Each involved object changes its corresponding state and updates its rate constants. When a new multi-state molecule is produced, a new object is created into the system. When an existing multi-state molecule degrades, the corresponding object is eliminated from the system. Thus the numbers of species and reaction channels vary. For certain systems, the dynamical changes in the numbers of species and reaction channels may be frequent. Such examples can be found

in [31] where the chemical network is updated dynamically and the standard SSA is applied to the dynamically varying chemical network. If the multi-state species are generated and eliminated with small populations, the HSSA strategy will show good efficiency.

Let us consider Example 2 again. Assume that the total population of E is E_T. As discussed before, the standard SSA will have 1002 species and about 3000 reaction channels. However, if we apply the HSSA method, each E molecule is treated as a multi-state object. For each E object the three reaction channels (2.24-2.26) have local copies. Thus $\mathcal{N}_{\mathcal{S}} = 2 + E_T$ and $\mathcal{M} = 3E_T$. When E_T is small, the efficiency gain is great. When E_T increases, the efficiency gain decreases. When E_T reaches around $1,000$, it will be less efficient than the standard SSA.

2.5 Numerical Experiments

In this section we present numerical experiments for the comparison of the SSA and StochSim.

2.5.1 Bacteria Chemotaxis Model

The first example we use is the bacteria chemotaxis model on which StochSim has been applied successfully. This model contains 12 reaction channels (see Section 5.2) with the assumption that some fast reaction channels always remain in equilibrium. The enzyme $TTWWAA$ plays an important role in this model. It has multiple states and can participate in various types of reactions such as phosphorylation, methylation, and binding with other chemicals. StochSim models each molecule of the enzyme $TTWWAA$ as an object with many different states. To implement the SSA, we have to transform different states of the enzyme $TTWWAA$ into different species. Each species corresponds to one state of the enzyme. By doing so, we increase the number of reactants and reactions in the system. The resulted model (see Section 5.2) contains 28 reaction channels.

Figure 2.2 shows the trajectories of the active enzyme $TTWWAA$ generated by SSA and StochSim respectively. We see that the trajectories match well with each other. The difference is mostly due to the fluctuation inherent with the stochastic simulation.

We listed the means and variances of the total population of the active $TTWWAA$ in Table 2.1. The corresponding histograms are shown in Figure 2.3. We can see that the results from StochSim and the SSA are very close. But the average tau value of SSA is 217 times larger than the Δt for StochSim, while the simulation time of SSA is 31 times faster than that of StochSim. Note that here for the fair comparison, StochSim has been rewritten in C language to improve the efficiency. For the original StochSim package, $10,000$ simulations took $219,280$ seconds CPU time, which is 5.6 times slower than our simple implementation in C language.

	Mean	*Variance*	*Simulation time*	*Average Stepsize*
StochSim	210.82	202.83	*39,202s*	3.6×10^{-5}
SSA	210.12	207.82	*1,268s*	7.9×10^{-3}

Table 2.1: Comparison of the means, variances, CPU times and average step sizes by the SSA and StochSim on the chemotaxis model for 10,000 runs.

2.5.2 Efficiency Comparison on a Simple Example

We implemented StochSim, the SSA and the HSSA methods on a modified version of Example 3. The reactions are listed below

$$\begin{aligned} E_n + X &\xrightarrow{k_1} E_n + Y \\ Y &\xrightarrow{k_2} X \\ E_n &\xrightarrow{k_3} E_{n+1} \\ E_n &\xrightarrow{k_4} E_{n-1} \end{aligned} \tag{2.27}$$

where E_n represents the state n in species E and $1 \leq n \leq 1,000$. The reaction rate $k_1 = 5 \times 10^3 n$ [2], where n is the index of the corresponding state of E. $k_2 = 1$ and $k_3 = k_4 = 0.1$.

[2] k_1 is much larger than other reaction rates because it is a bi-molecule reaction rate.

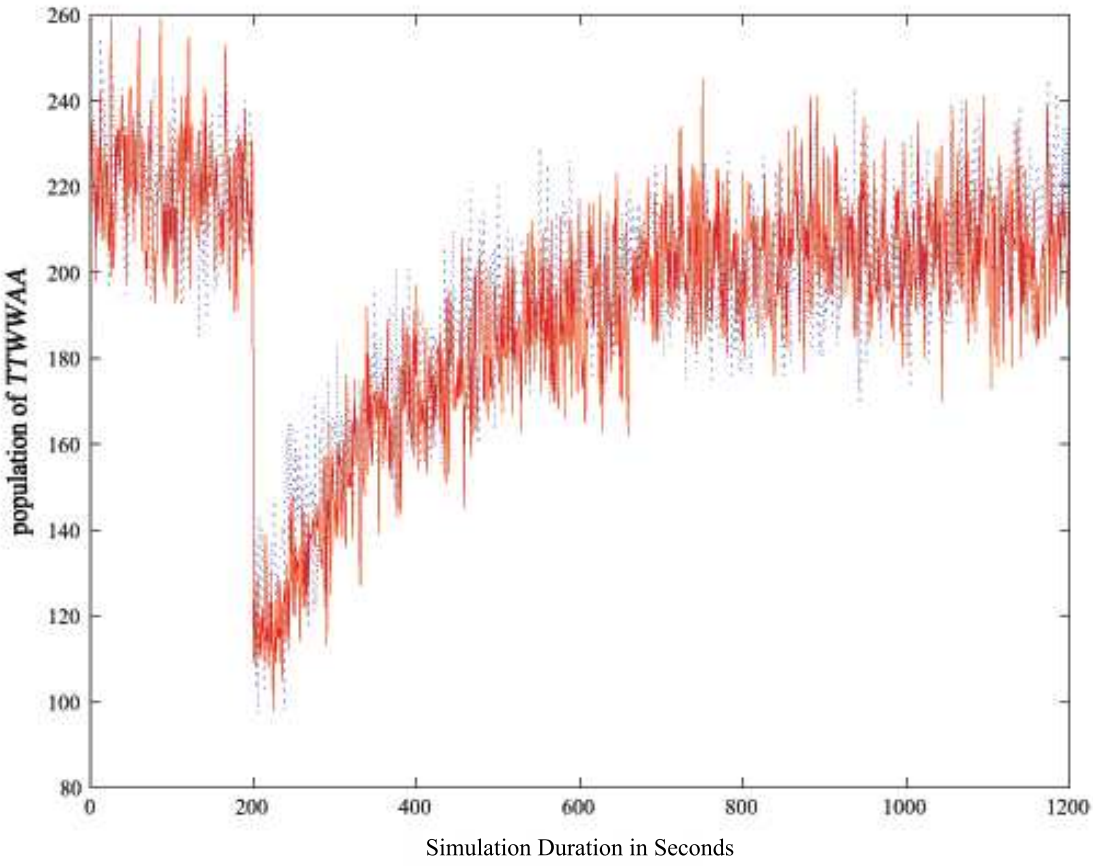

Figure 2.2: The solid red trajectory is simulated by the SSA, and the dotted blue one is by StochSim. At the time of 200s, there is a sudden drop because a certain amount of aspartate is added into the system, repressing the activation of $TTWWAA$.

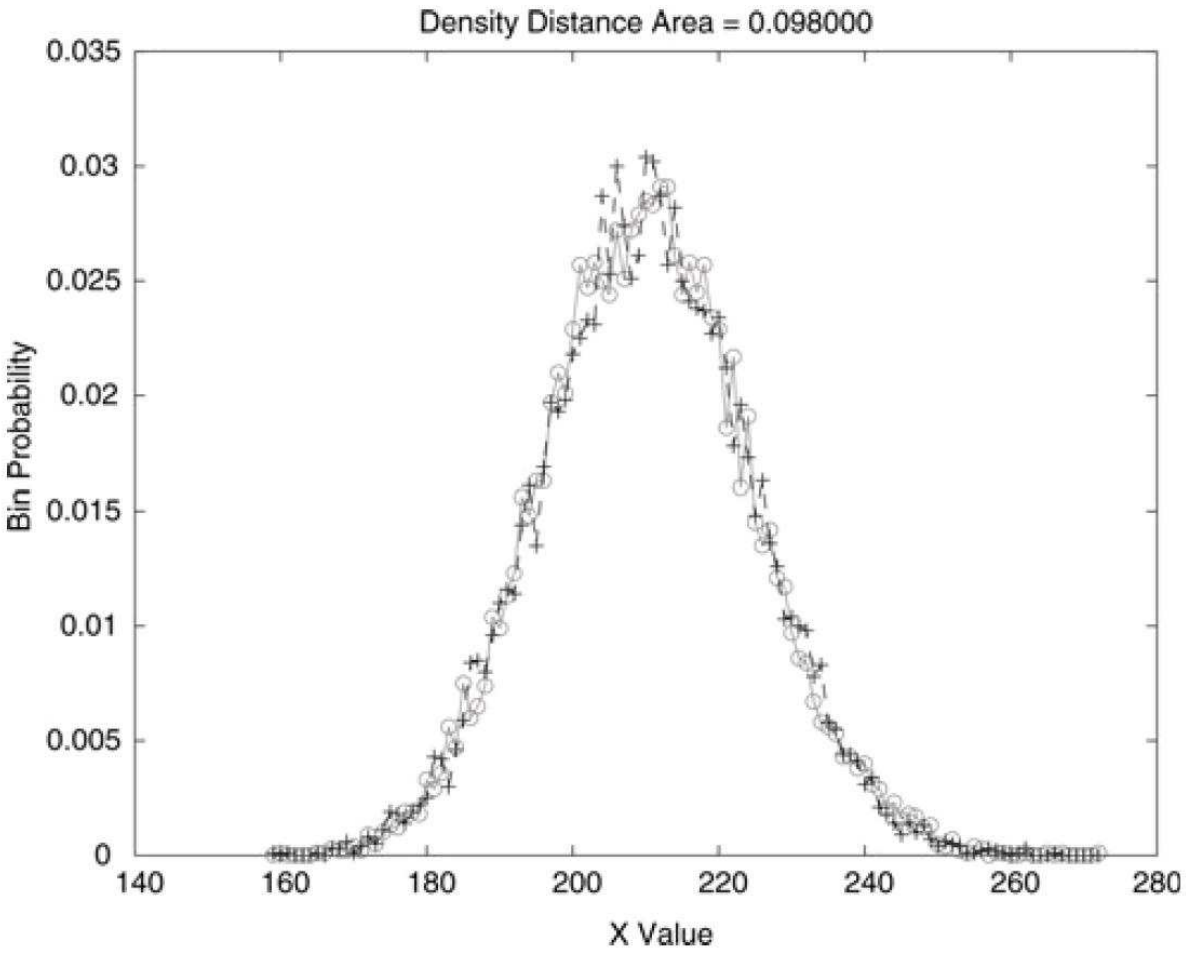

Figure 2.3: The histograms of the active TTWWAA simulated by the SSA(grey) and StochSim(black).

First we fix the initial population of $X = 100$ and $Y = 0$ and increase E_T, the population of E, from 1 to $10,000$.

The CPU time for different methods are listed in Table 2.2. For this simple example, when the population of E is small, the HSSA is the most efficient, and StochSim is a little more efficient than the SSA. As E_T increases by a factor of 10, the CPU time for the SSA increases with a factor between 6 and 10. For StochSim, when $E_T < 100$, the CPU time increased little. After $E_T > 100$, the CPU time shows super-linear increase. The CPU time of the HSSA shows even more obvious super-linear increase trend. As E_T becomes large, the HSSA becomes the slowest method.

Population of E	*1*	*10*	*100*	*1,000*	*10,000*
SSA	*0.81*	*6.93*	*58.29*	*324*	*2,013*
StochSim	*0.31*	*0.34*	*0.90*	*11.42*	*666*
HSSA	*0.013*	*0.11*	*6.64*	*425*	*22,122*

Table 2.2: Comparison of CPU times in seconds among the SSA, StochSim, and the HSSA for 1,000 runs of the model (2.27)

The simulation results can be explained with complexity analysis. The CPU time is decided by the computational cost in a single step multiplied by the total number of steps. For SSA, as E_T increases, the propensity values $a_j(x) = O(E_T)$ except the second reaction channel in (2.27). The computational cost in a single step of the SSA does not change with E_T. The number of steps is of order $O(\frac{1}{\tau})$. Because the τ value is of order $O(\frac{1}{a_0})$, the number of total simulation steps is then of order $O(a_0)$. But for this example, $a_0(x) = \sum_j a_j(x)$ is of order $O(E_T)$. Thus the CPU time is of order $O(E_T)$. For StochSim, when E_T increases, the computational cost in a single step does not change much. The CPU time is proportional to $\frac{1}{\Delta t}$. From (2.6) we know that the CPU time is of order $O(N^2)$, where N is the total population in the system. In this example, $N = 100 + E_T$. When $E_T < 100$, N does not change much as E_T increases. Thus the CPU time does not increase much. But when $E_T > 100$, N increases linearly with E_T. Thus the CPU time increases quadratically with

E_T. For the HSSA, since it is still the SSA method, the number of steps should be the same as the situation in the standard SSA. Thus the number of the steps for the HSSA is of order $O(E_T)$. However, its computational cost in a single step is $O(\mathcal{M})$. When E_T is small, $\mathcal{M}$ is small. This cost is small. When E_T increases, each new object will have its own copy of reaction channels. We have $\mathcal{M} = O(E_T)$. Thus the computational cost in a single step is $O(E_T)$. Multiplying the computational cost in a single step and the total number of steps, we know that the CPU time for the HSSA is of order $O(E_T^2)$.

From this example we can see that the HSSA works well when E_T is small. When the total population increases, the efficiency of StochSim drops quadratically. But with a large E_T, StochSim still shows advantages for systems involved with multi-state species. Further improvement of HSSA is in need to combine the advantages of StochSim and the SSA.

Chapter 3

A Hybrid Strategy Combining Improved StochSim and SSA

The HSSA presented in the previous chapter shows a high efficiency when the number of multi-state molecules is small. However, when the number of multi-state molecules increases, the large numbers of species and reaction channels still present a challenge. A good solution is still in demand. In this chapter, we propose a hybrid scheme that effectively combines the advantages of StochSim and the SSA. Then results of numerical experiments are provided.

3.1 A Hybrid Method

From the comparison between the SSA and StochSim we know the following results: SSA has advantage in simulating systems with large populations, while StochSim has the capability to overcome the multi-state obstacle [24]. It is easy to choose one algorithm over the other if the problem is well-defined. However, often we have complicated models with high populations of different species of molecules, at the same time containing one or more species of multi-state complex molecules. In this case, we need a hybrid scheme to combine the advantages of

both the SSA and StochSim and overcome each algorithm's drawback. This hybrid scheme should exhibit higher efficiency than either of the SSA and StochSim.

To achieve this goal, we must notice that the major difference between the two methods is that StochSim takes a fixed and very small step size while the SSA selects an adaptive one in each step. As a result, the combination of the two requires StochSim use a new strategy to generate its step size.

3.1.1 An Improved StochSim

To facilitate the combination of the SSA and StochSim, we propose an improved version of StochSim with modifications in two aspects to avoid generating a fixed and too small step size.

Adaptive Step Size

In StochSim. the total number of molecules N is maximized and fixed. The step size Δt is also fixed before simulation by (2.6), and approximately proportional to the inverse of N^2. For many realistic systems the actual total number of molecules in system may change dynamically. Taking the maximal value for N can result in a very small fixed Δt, thus a low efficiency. Therefore, it is desirable to have an adaptive step size Δt, varying with the actual total number of molecules in the system.

In our improved version of StochSim, for those models whose total number of molecules may change drastically over time, we modify the algorithm as follows. First, no look-up table is needed any more. In the simulation, at each time step, use N, the actual total number of molecules in the system, to calculate Δt using (2.6). Then when molecules are selected, the probability for the corresponding reaction is calculated using (2.4) or (2.5).

Eliminating Pseudo-Molecule

The original StochSim uses pseudo-molecule objects to differentiate uni-molecule reactions from bi-molecule reactions. Although having this definition does facilitate deriving the probability formula, it adds redundant information into the system. We have derived an improved algorithm that does not need pseudo-molecules. We will show that this change makes the method more efficient.

To avoid pseudo-molecules, we further modify StochSim based on the improved version described in the previous section. In initialization, we no longer calculate N_0 or create any pseudo-molecule. In each time step, we use a different formula to optimize Δt, and different formula to calculate the probability for the possible reaction. The whole simulation procedure is given as the following:

1. Calculate the actual number of molecules in system. Calculate adaptive Δt with (3.3).

2. Randomly select the first reactant from all molecules, using uniform distribution.

3. Search for possible uni-molecule reactions for the selected molecule. If no corresponding reaction is found, go to step 4. Otherwise, calculate the corresponding possibility using (3.1) and compare it with a uniform random number generated in $(0, 1)$. If the random number is larger, go to step 4. Otherwise there is a uni-molecule reaction to fire. If there is only one possible reaction channel for the selected molecule, select this reaction and go to step 6. Otherwise, the code selects one of the possible reaction channels in a similar way as the equation (2.2) in the SSA and go to step 6.

4. Randomly select the second reactant from all molecules, using uniform distribution.

5. Search possible bi-molecule reactions between the two selected objects. If no corresponding reaction is found, StochSim concludes that no reaction occurs in this time step. Otherwise, calculate the corresponding possibility using (3.2) and compare it with a uniform random number generated in $(0, 1)$. If the random number is larger,

StochSim concludes that no reaction occurs in this time step. Otherwise there is a bi-molecule reaction to fire. If there is only one possible reaction channel between these two molecules, the reaction is selected to fire. Otherwise, the code selects one of the possible reaction channels in a similar way as the equation (2.2) in the SSA.

6. Update the system accordingly if any reaction channel is selected. Proceed to the next time step.

By using the modified procedure above, the advantages are at least in two respects. Firstly, we do not have to introduce pseudo-molecules into the system, thus simplifying algorithm implementation. Secondly, since the simulation procedure has been different and we do not have N_0 any more, the possibility formula (2.4) and (2.5) have been replaced by (3.1) and (3.2) as follows. For a uni-molecule reaction,

$$p'_{1j} = k_{1j} N \Delta t. \tag{3.1}$$

For a bi-molecule reaction,

$$p'_{2j} = \frac{k_{2j} N(N-1)\Delta t}{2 N_A V}. \tag{3.2}$$

Consequently, formula (2.6) has also been changed as follows.

$$\Delta t = min(\frac{MAXP}{k_{1max} N}, \frac{MAXP}{\frac{k_{2max} N(N-1)}{2 N_A V}}). \tag{3.3}$$

As we can see that, the optimized Δt calculated from formula (3.3) is larger than the one derived from formula (2.6), thus making the algorithm more efficient.

However, one may argue about the inaccuracy caused by these modified formula. In step 3 and (3.1), the method actually assumes that, if any uni-molecule reaction were to fire, then no bi-molecule reactions would be considered to fire. Then this assumption would introduce error. Therefore, the possibility calculated from (3.2) is actually not precise. This is true, and the accurate form for bi-molecule reaction j should be:

$$p''_{2j} = p'_{2j}(1 - p'_{1j}) = p'_{2j} + O((\Delta t)^2), \tag{3.4}$$

where p'_{1j} is the possibility calculated in step 3, and p'_{2j} is the possibility calculated in step 5. As we can see, the difference between (3.2) and (3.4) is $O((\Delta t)^2)$. According to the calculation shown in (2.9), the error caused by (3.4) is in the same order as the error of the original StochSim is, therefore it will not cause loss of accuracy.

3.1.2 Combination of SSA and StochSim

With the improved generation strategy of the time step size, we are able to combine StochSim with the SSA. Our main idea is to partition the complicated system into two subsystems, one for SSA and the other for StochSim. The whole reaction set is partitioned into two subsystems, according to the types of the reactants in each reaction. Suppose the system has N_n non-multi-state species $\{Sn_1, \ldots, Sn_{N_n}\}$, and N_m multi-state species $\{Sm_1, \ldots, Sm_{N_m}\}$. The system can involve three types of reaction channels:

1. *nn reactions*: uni-molecule reactions for one non-multi-state species and bi-molecule reactions between two non-multi-state species;

2. *mn reactions*: uni-molecule reaction for one multi-state species and bi-molecule reactions between one non-multi-state-species and one multi-state species;

3. *mm reactions*: bi-molecule reactions between two multi-state species.

Before simulation, the system is divided into the two following subsystems in terms of the descriptions below:

1. The SSA subsystem: this subsystem contains all nn reactions, with only non-multi-state species as reactants. This subsystem can be simulated by the SSA, which keeps a state vector of populations for all non-multi-state species.

2. StochSim subsystem: this subsystem contains all reactions involved with at least one multi-state species, namely all mn reactions and mm reactions. This subsystem can be

simulated by our improved version of StochSim, in which the state vector represents the molecules.

After the division, the hybrid scheme follows the principle of the first reaction method version of SSA, to find *the next occurring reaction*. Basically, in each time step, each subsystem generates a time at which its next reaction will occur. The two times are compared to figure out which reaction in which subsystem should fire first. However, we must note that, StochSim subsystem does not actually predict the time in which the next reaction will occur, but for a time interval it decides whether or not there is a reaction firing.

Suppose current time is t. If the SSA subsystem time denoted by τ is smaller than StochSim subsystem time denoted by Δt, it is important to figure out in time interval $[t, t+\tau)$ whether there is a reaction to fire in StochSim subsystem, because no reaction will fire in the SSA subsystem in this interval. Then we proceed StochSim subsystem with a step size of τ instead of Δt, knowing that reducing Δt does not harm the accuracy of StochSim. If there is a reaction to fire in StochSim subsystem, then it is the next occurring reaction. Otherwise the reaction selected in the SSA subsystem at time $t + \tau$ is the next occurring reaction.

If τ is larger than or equal to Δt, we should figure out in time interval $[t, t + \Delta t)$ whether there is a reaction firing in StochSim subsystem. Then we proceed StochSim subsystem with a step size of Δt. If there is a reaction to fire, then it is the next occurring reaction. Otherwise, let $\tau = \tau - \Delta t$, and redo the comparison between τ and Δt to continue looking for the next occurring reaction.

Therefore, we have the simulation proceeds as follows:

1. The SSA subsystem generates τ.
2. StochSim subsystem generates an adaptive step size Δt.
3. Compare τ with Δt.
4. If τ is smaller, proceed StochSim subsystem with a step size of τ and let $t = t + \tau$. If

any reaction occurs, go back to step 1. Otherwise, generate j, fire Reaction j in the SSA subsystem, and go back to step 1.

5. If τ is larger, proceed StochSim subsystem with a step size of Δt and let $t = t + \Delta t$. If any reaction occurs, go to step 1. Otherwise, let $\tau = \tau - \Delta t$ and go to step 3.

The procedure is also illustrated in Figure 3.1.

By dividing the system into two parts, the hybrid method actually prevents the SSA subsystem from dealing with multi-state species, meanwhile preventing StochSim subsystem from dealing with non-multi-state species with usually high populations. In this way, both subsystems take advantages of their own and avoid their drawbacks.

3.2 Numerical Experiments

To test the accuracy and efficiency of the hybrid method, we have constructed a toy model with several non-multi-state species and one multi-state species. Simulations are done with a Linux workstation based on Intel Pentium 4 CPU of 2.80GHz. The system is composed of the following reactions:

$$A + C \rightleftharpoons D \rightarrow P$$

$$B + C \rightleftharpoons E \rightarrow Q$$

$$A \overset{F_n}{\rightleftharpoons} B$$

$$F_n \rightleftharpoons F_{n-1}$$

$$F_n \underset{Q}{\overset{P}{\rightleftharpoons}} F_{n-1}$$

where F is the multi-state species and F_n represents F in state n. Consider F as an enzyme with a number of different states and each state has a different reaction rate constant for its enzyme ability. In this model, the rate constant is linearly related to the state number. To

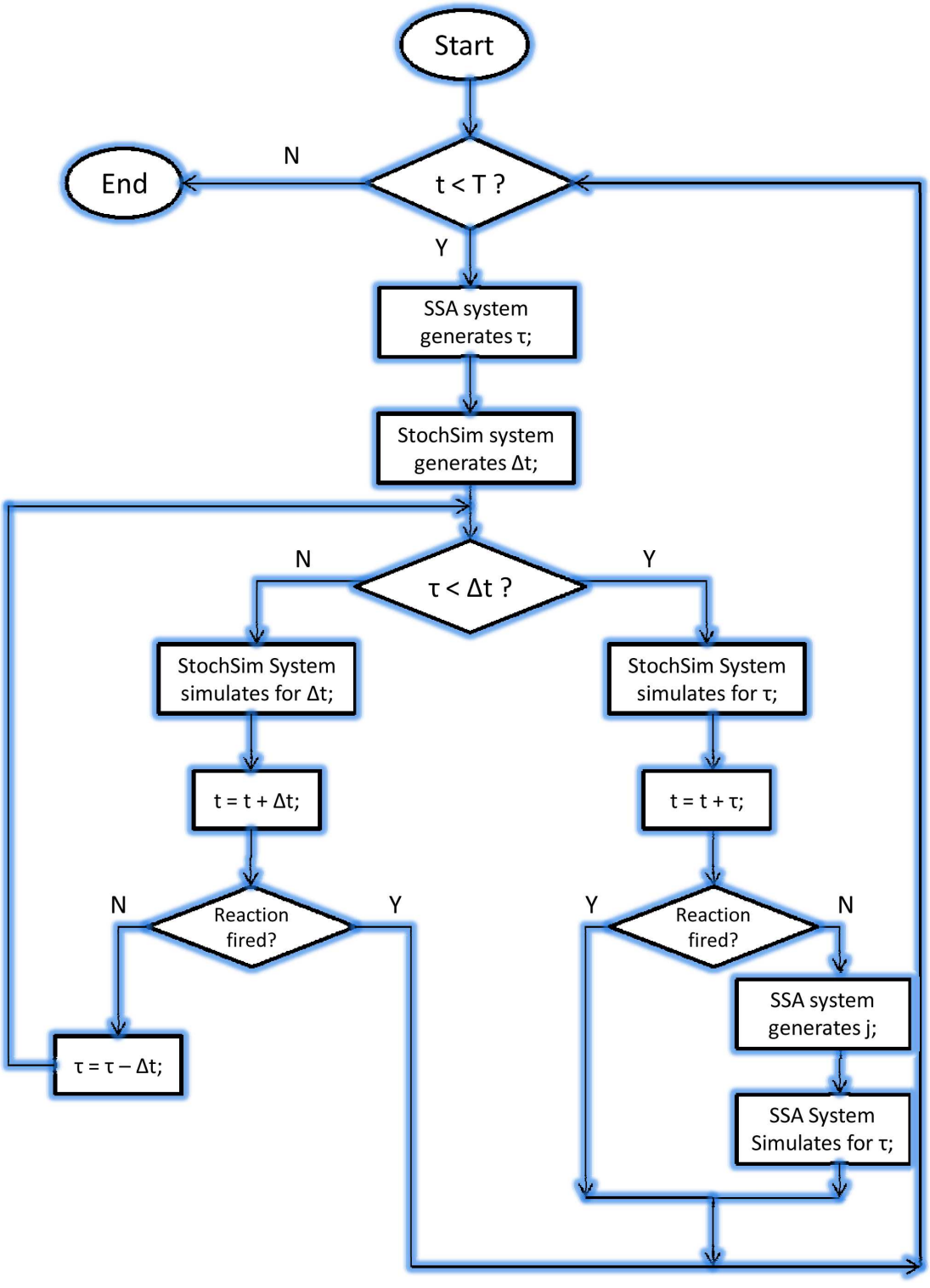

Figure 3.1: Flowchart of simulation procedure of each time step in the hybrid method.

simulate this system with the hybrid scheme, all reactions involved with F are considered mn reactions, thus included in StochSim subsystem, and all other reactions are nn reactions, included in the SSA subsystem. This model does not contain any mm reaction yet.

We analyze the efficiency of the hybrid method, the SSA and StochSim from three perspectives to see how different features of the system affect the performance. In three groups of experiments, we respectively vary the populations of all the non-multi-state species, the number of the states of F and the population of F. The results are shown in Table 1-3. We can see that the hybrid scheme generally succeeds in combining the advantages of the SSA and StochSim and overrunning both of them in all situations.

Initial Populations of S_{nms}	1	10	100	1,000	10,000
Hybrid	0.266	0.358	1.164	10.125	100.93
SSA	2.732	3.481	5.208	121.96	1217.9
StochSim	0.249	0.508	7.377	370.61	5082.4

Table 3.1: S_{nms} indicates all non-multi-state species including A, B, C, D, E, P and Q. The numbers in the table are simulation seconds for 1,000 runs on the model above by different algorithms.

Number of States of F	3	10	100	1,000	10,000
Hybrid	1.127	1.160	1.264	1.155	1.163
SSA	1.329	2.045	13.127	123.76	1339.2
StochSim	6.769	7.035	6.845	7.010	7.718

Table 3.2: The numbers in the table are simulation seconds for 1,000 runs on the model above by different algorithms.

Then we modify the model by replacing the reverse reaction in the last row

$$F_n \xrightarrow{Q} F_{n+1}$$

with the two following reactions

$$F_n \xrightarrow{G} F_{n+1}$$

$$G_n \xrightarrow{Q} G_{n+1},$$

Population of F	1	10	100	1,000	10,000
Hybrid	0.580	0.578	1.125	5.813	52.163
SSA	5.857	6.362	13.062	68.589	596.53
StochSim	5.300	5.228	6.692	22.730	637.98

Table 3.3: The numbers in the table are simulation seconds for 1,000 runs on the model above by different algorithms.

where G is also a multi-state species with a state of n. Note that the second reaction is an mm reaction. Similar experiments were conducted and we obtained similar results as above.

Numerical experiment results demonstrate that the new method has much better performance than either of the original algorithms for systems with both multi-state species and a large total population.

Chapter 4

Rule-Based Modeling and Simulation

Besides StochSim, another way to deal with the multi-state situation is to use rule-based modeling [11]. In this chapter, we first introduce the idea of this modeling method and one of its simulation methods called the Network-Free Algorithm. Then, we propose two new methods for simulating rule-based models and numerical results are provided.

4.1 Background

4.1.1 Rule-Based Modeling

The rule-based modeling uses rules to replace reaction channels. Here a rule defines how a particle may react with other particles, depending on the states of their binding sites. Suppose protein species A has 3 binding sites and the first site can either be open or occupied by a protein in species B. One rule can include all reactions that bind a B protein onto the first site of an A protein, regardless of the other 2 sites. Such a rule can be expressed as

$$A_{open,?,?} + B \xrightarrow{k_{rule}} A_{B,?,?} \tag{4.1}$$

where the subscripts of a species describe the situation of all the binding sites. A question mark indicates all possible states for a certain site. k_{rule} is the rule rate constant. However, to represent this single rule, the SSA must assign one reaction to every qualified state variables of species A and may have a large system size.

In BioNetGen [26], this rule can also be equivalently represented as

$$A_{open,?,?} + B_{open} \xrightarrow{k_{rule}} A_{B,?,?} + B_A \tag{4.2}$$

The rule-based modeling is particularly convenient for representing biochemical systems with complex chemicals that may bind to each other. Consider the following example. Suppose there is only one species with three binding sites, and each empty binding site can be bound to an empty binding site of another particle from the same species. When these two particles bind together, the remaining empty sites may still be bound by other particles. So eventually a complex may have many particles which collectively have thousands of empty binding sites remaining. Rule-based modeling was proposed to represent this type of systems which apparently may have a huge or even infinite number of reactions, but follows very simple rules.

4.1.2 Network-Free Algorithm

Due to combinatorial complexity, the rule-based models usually have a very large system size if represented with the SSA. Recently, a new particle-based method called Network-Free Algorithm (NFA) [29] was proposed to handle the combinatorial complexity. In the NFA, the propensities are for rules and calculated in the similar way as in the SSA. The only difference is that, instead of using populations of the reactant species in reactions, it uses populations of all the particles that are candidates of the reactants of rules. For example, for the bi-molecule rule (4.1), its propensity is calculated by the reaction rate k_{rule} times the number of molecules in species A with the first site open, then times the number of molecules in species B. The time and index for the next reaction are still calculated as in (2.1) and

(2.2). Then molecules of reactants in the selected rule are randomly picked up. The general procedure of the NFA for our implementation is as follows.

Initialization

In general, every rule has the 4 following properties.

1. Descriptions of the reactants and products with their species and states of all the binding sites.
2. Rule rate constant.
3. Lists of candidate reactants. To facilitate selecting reacting particles after the rule index is determined, the method maintains a reactant list for each reactant in each rule. A reactant list contains pointers to all the candidate particles qualifying for the corresponding reactant descriptions in its rule.
4. Populations of reactants. To calculate propensities in a fast manner, populations of reactants are recorded so that counting the particles in reactant lists is not repeated in every step.

The algorithm first creates rules by initializing their first two properties. Then objects for all molecule particles in the system are created. Upon the creation of each object, all the rules are scanned. If the created object qualifies for one reactant of one rule, the pointer of the object is added to the corresponding reactant list and the reactant population is updated. Therefore, after all the particle objects are created, the initialization of the last two properties of the rules are completed.

Simulation

In simulation, the NFA repeats the simulation procedure as follows until a stopping criterion is reached.

1. Calculate propensities for all rules in the same way as in the SSA, using the rules' reactant populations and rate constant.
2. Calculate the next event time with (2.1).
3. Select a rule with (2.2).
4. Select reacting objects from corresponding reactant lists of the selected rule using uniform random numbers.
5. Remove pointers of the reacting objects from their reactant lists of the selected rule, and decrease corresponding reactant populations by 1.
6. Destroy, create or change reacting objects according to the descriptions of the selected rule.
7. For each product object, scan all the rules. If the product object qualifies for one reactant of one rule, the pointer of the object is added to the corresponding reactant list and the reactant population is increased by 1.
8. Update time, and go to Step 1.

The advantages of this method are that, it does not generate null events as StochSim often does, and it does not need to generate the huge reaction network as the SSA does. The disadvantage is the extra cost on updating the reactant lists and populations of all the rules in every time step.

4.2 Population NFA (PNFA)

Since the NFA creates objects for all the particles and keeps reactant lists for each rule, if the total population in the system is very large, the method can cause severe burden in memory allocation and greater cost in maintenance of those very long reactant lists. In these cases, we

propose an improved NFA which treats all non-multi-state species with populations instead of creating individual particles for them. In other words, the improved NFA creates only one object for each non-multi-state species. Objects of this type are called the *population objects*. The only difference between population objects and normal objects is that the former objects contain the populations of the species instead of states of binding sites. With this change, the space complexity of NFA is greatly reduced for those systems involving simple chemical species with high populations. The procedure of PNFA is as follows.

Initialization

The algorithm first creates rules by initializing their first two properties. Then objects for all molecule particles in the system are created species by species. For a non-multi-state species, a species object is created with its initial population. For a multi-state species, normal particle objects in the number of its population are created with binding site states. Upon the creation of each object, all the rules are scanned. If the created object qualifies for one reactant of one rule, the pointer of the object is added to the corresponding reactant list and the reactant population is increased by 1. After all the objects are created, the initialization of the last two properties of the rules are completed.

Simulation

In simulation, repeat the simulation procedure as follows until a stopping criterion is reached.

1. Calculate propensities for all rules in the same way as in the SSA, using the rules' reactant populations and rate constant.
2. Calculate the next event time with (2.1).
3. Select a rule with (2.2).

4. Select reacting objects from corresponding reactant lists of the selected rule using uniform random numbers. During the selection, all population objects are considered normal objects in the numbers of their populations.

5. For normal reactant objects, remove their pointers from their reactant lists of the selected rule. Then for all reactant objects, decrease corresponding reactant populations by 1.

6. Destroy, create or change reacting objects according to the descriptions of the selected rule.

7. For each product object, scan all the rules. If the product object qualifies for one reactant of one rule, for normal product object its pointer is added to the corresponding reactant list; then for all product objects the reactant population is increased by 1.

8. Update time, and go to Step 1.

4.3 Full-Scale SSA (FSSSA)

The PNFA is in itself a hybrid scheme combining both population-based and particle-based strategies. Then some questions naturally arises: Is the particle-based scheme the only way to simulate rule-based models? Is there any case where a purebred population-based method performs the best? To our estimation, besides non-multi-state species, any multi-state species with a large population and a small number of states should suit more for the population-based scheme. On the other hand, particle-based scheme is preferred when a small population of a multi-state species is sparsely distributed into a large number of states.

To further explore for the best simulation method, we propose the Full-Scale SSA, a population-based algorithms for simulating rule-based models, further derived from the PNFA. Because of its population-based nature, it can also be considered as another version of the SSA. Below is its simulation procedure.

Initialization

Like the NFA, rules are first initialized with their first two properties. Since the method is no longer particle-based, it does not create objects for molecules and the react lists for rules are not needed either. Species are initialized one by one. A non-multi-state species is considered a multi-state species with one state. For each species a state variable matrix is created for all the possible combinatorial states, and is initialized state by state. Upon initialization of each state, its state variable is assigned with its population, and all the rules are scanned. If the state variable qualifies for one reactant of one rule, the corresponding reactant population is increased by its population.

Simulation

In simulation, repeat the simulation procedure as follows until a stopping criterion is reached.

1. Calculate propensities for all rules in the same way as in the SSA, using the rules' reactant populations and rate constant.
2. Calculate the next event time with (2.1).
3. Select a rule with (2.2).
4. Randomly selecting reacting state variables from the corresponding reactant lists of the selected rule. The selection procedure is the same as in (2.2).
5. Update reacting state variables according to the descriptions of the selected rule.
6. Scan all the rules. If the reactant variable qualifies for one reactant of one rule, the corresponding reactant population is decreased by 1. If the product object qualifies for one reactant of one rule, the corresponding reactant population is increased by 1.
7. Update time, and go to Step 1.

4.4 Comparison of Simulation Methods

So far seven simulation methods have been analyzed. Suppose N_R is the number of rules in the system, N_S is the maximal number of state for one multi-state species, and N_{obj} is the total number of objects in the system. To better understand the similarities and differences of all these methods, we provide a brief comparison in their simulation natures and time/space complexities as in Table 4.1.

	Simulation	*Time*	*Time Complexity in Each Step*		*Space*
	Scheme	*Steps*	*reactant selection*	*system update*	*Complexity*
SSA	*population-based*	$O(a_0)$	$O(log(\mathcal{M}))$	$O(1)$	$\mathcal{N_S}$
StochSim	*particle-based*	$O(N^2)$	$O(1)$	$O(1)$	$O(N_{obj})$
HSSA	*population-based*	$O(a_0)$	$O(log(\mathcal{M}))$	$O(1)$	$\mathcal{N_S}$
Hybrid	*hybrid*	*N/A*	$O(log(\mathcal{M}))$	$O(1)$	$O(N_n + N_{obj})$
NFA	*particle-based*	$O(a_0)$	$O(log(N_R))$	$O(N_R)$	$O(N_{obj})$
PNFA	*hybrid*	$O(a_0)$	$O(log(N_R))$	$O(N_R)$	$O(N_{obj})$
FSSSA	*population-based*	$O(a_0)$	$O(log(N_R) + log(N_S))$	$O(N_R)$	$\mathcal{N_S}$

Table 4.1: Comparison of all methods.

Here we only focus on comparison among the NFA, PNFA and the FSSSA. we first compare the NFA and PNFA. Since they both use uniform random numbers to select reactant objects in constant time, the first parts of their time complexities are both dominated by selecting the rule to fire, which are $O(log(N_R))$. But for the system update part, since there is good chance for the PNFA to select population objects and not have to update corresponding reactant pointer lists, the PNFA often has a smaller co-efficient for $O(N_R)$. So the total time complexity of the PNFA is often lower than the NFA. Also, the space complexity of the PNFA is often lower because of the population objects, which can greatly reduce N_{obj}.

Now between the PNFA and FSSSA, since the latter still uses (2.2) to select the states of the reacting variables from multi-state arrays, the first part of its time complexity has an

extra term of $log(N_S)$. However, for the system update part, the FSSSA does not maintain reactant pointer lists, so its co-efficient of $O(N_R)$ can be even smaller than that of the PNFA. Therefore, when N_S for the FSSSA is not large but N_{obj} for the PNFA is large, the FSSSA can exhibit a lower total time complexity. The space complexity of the FSSSA is the same as that of the SSA because both of them have the same number of state variables. However, it is obvious the FSSSA consumes the least memory space among all the methods listed in Table 4.1.

4.5 Numerical Experiments

To compare both the accuracy and efficiency of the PNFA and FSSSA, we constructed a simple toy model containing two reversible rules as the following:

$$A \overset{S_n}{\rightleftharpoons} \phi$$

$$S_n \overset{A}{\rightleftharpoons} S_{n+1}$$

In this model, ϕ stands for null species, and S is a multi-state species with 8 states representing 8 different phosphorylation levels. S_n denotes species S in state n. S acts as an enzyme for the degradation of A, but its activity is inversely related to its state number. Therefore, molecules of S with a low phosphorylation level tend to degrade A while highly phosphorylated S molecules activate A. On the other hand, A is also an enzyme that helps the phosphorylation of S. Thus these two rules form a positive feedback loop.

This model was simulated for 10,000 runs on an iMac Core 2 Duo 2.0GHz machine, with both the PNFA and FSSSA. The whole process took the FSSSA 684.1 seconds while it cost the PNFA 2387.1 seconds. This efficiency difference can be explained by our previous complexity analysis in Table 4.1. Based on these 10,000 runs by both methods, Table 4.2 shows the population distributions of S on all of the states. This figure demonstrates the bi-stability

nature of this model caused by the positive feedback loop. We could see the mean values of two methods are very close to each other. Figure 4.1 also shows the comparison of the

State Number	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
PNFA	*541.1*	*910.0*	*1522.1*	*2027.9*	*2028.1*	*1521.9*	*909.8*	*539.2*
FSSSA	*541.0*	*910.4*	*1522.7*	*2027.9*	*2027.4*	*1521.9*	*909.6*	*539.0*

Table 4.2: Population distribution on states of S. The numbers in the table are the mean values for 10,000 runs.

histograms of S_0 simulated for 10,000 runs. We can see the results from both methods are very close. For additional experimental results of comparison among StochSim, PNFA and FSSSA, refer to Section 5.3.

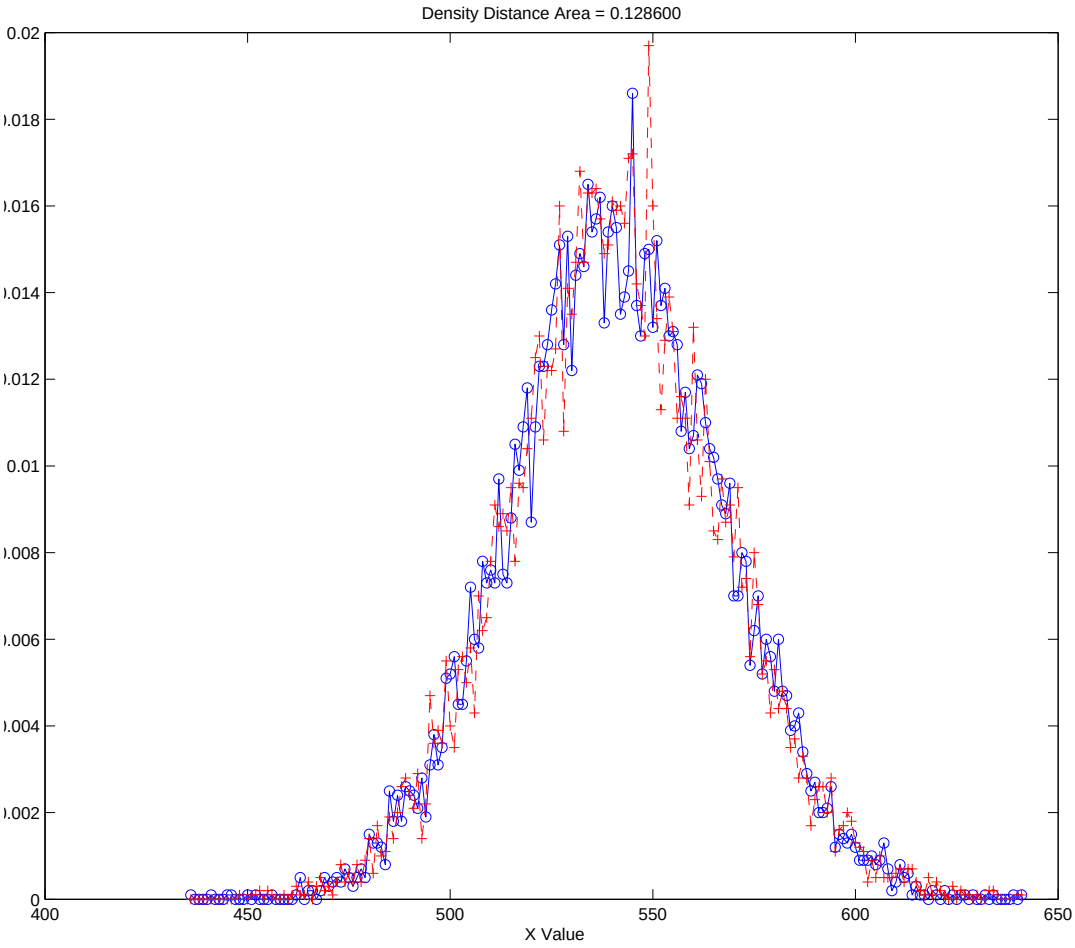

Figure 4.1: Histograms of S_0 simulated by PNFA (red) and FSSSA (blue).

Chapter 5

Biological Example: Bacterial Chemotaxis

In this chapter, we first introduce the bacterial chemotaxis in Section 1. In Section 2, three chemical models are provided. In Section3, we implement the PNFA and FSSSA on the complete model described in Section 2 and show the experiment results. In the last section, we describe our computer-based graphical interface for simulations of the bacterial chemotaxis, and present experimental results.

5.1 Background

The bacterial chemotaxis generally describes the phenomenon that bacteria respond to the chemical changes in their living environment by directing their movements. These changes include the fluctuation of both chemical attractants and repellents. By detecting such changes in their environment, some bacteria are enabled to swim towards the highest concentration of attractant chemicals such as food, and flee from poisonous chemicals.

Bacterial chemotaxis is achieved through a complicated chemical network. The mechanism

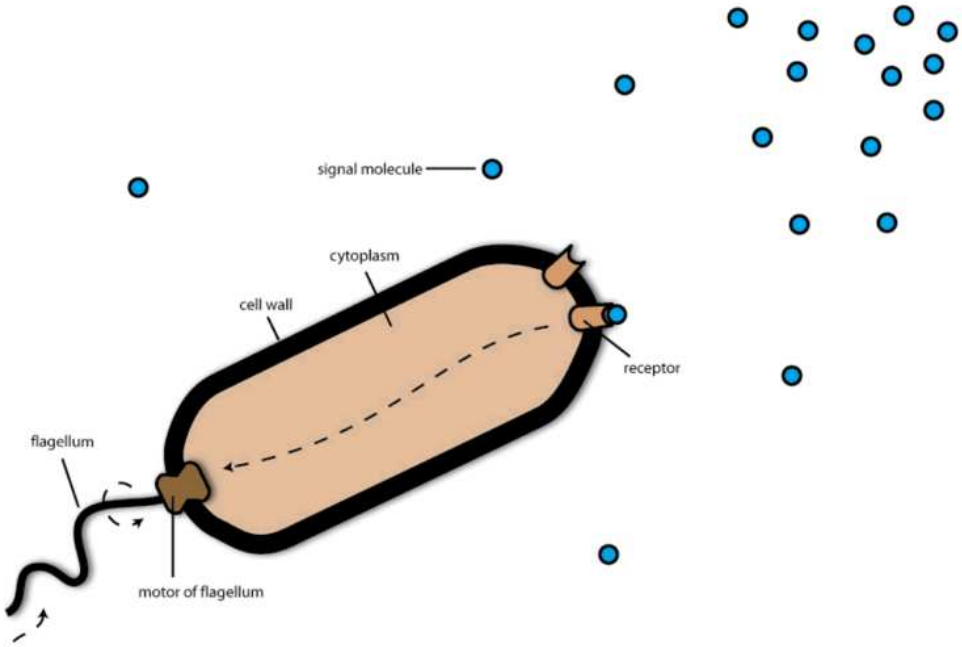

Figure 5.1: Bacterial chemotaxis mechanism [32].

of bacterial chemotaxis is shown in Figure 5.1 [13, 32]. At first, the receptors on the bacterial cell surface receive signals from certain environmental chemicals. Then the signals are passed into the cell, resulting in a series of chemical reactions, and finally regulate the level of the chemical *CheYp*, which controls the rotation of the motors. Lower concentration of *CheYp* tends to cause the bacterium's motors to rotate counter-clockwise. If all the motors rotate counter-clockwise, the flagella of the bacterium are bundled, and the bacterium is able to swim in a straight line (run mode). Otherwise, the entire flagella bundle can be interrupted by the direction change of a single motor from counter-clockwise to clockwise. Then the bacterium keeps changing direction after very short runs, and stays around (tumble mode). The run and tumble pattern is a special strategy of bacterial chemotaxis. Bacteria tumble when they cannot sense chemical concentration differences, and run otherwise. Biological experiments show that E. coli. is able to react with a slight chemical change as low as 1% [13].

5.2 Biochemical Models

The biochemical model of bacterial chemotaxis describes the chemically reacting system inside the cell. Figure 5.2 illustrates the system. The central enzyme $TTWWAA$ plays a key role in the model. This enzyme is composed of two molecules from each of *Tar*, *CheW* and *CheA*, and can be in an active or inactive form. Its activity can be enhanced when it is at a high methylation level, and when it is not bound to an aspartate molecule. Activated $TTWWAA$ can be auto-phosphorylated and then react with *CheY* and *CheB*, producing *CheYp* (phosphorylated *CheY*) and *CheBp* (phosphorylated *CheB*). Molecules of *CheYp* can bind to the motor complex of the cell flagella, increase the probability that the motor switches from counter-clockwise rotation to clockwise rotation, and finally decide the movement of the bacterium. *CheBp* and *CheR* are enzymes that regulate the methylation level of $TTWWAA$. *CheBp* binds to active $TTWWAA$ and assists removing methyl groups, while *CheR* binds to the inactive form and helps add methyl groups. Both enzymes form a negative feedback loop in the system.

The model that was first successfully simulated by StochSim is a simplified version [20]. It does not include *CheY*, *CheYp* and *CheB*, as well as the auto-phosphorylation of $TTWWAA$. The model takes the population of the active $TTWWAA$ instead of *CheYp* as output because it directly activates *CheYp* and should always exhibit the same behaviors as *CheYp*. After our modification, the model treats $TTWWAA$ as a multi-state species with 3 binding sites which respectively describe the following properties:

1. Bound to *CheR* (R), bound to *CheBp* (B_p), or not bound (N).
2. Level of methylation (0, 1, 2, 3, 4).
3. Active (A) or inactive (I).

Since the activation and deactivation of $TTWWAA$ are very fast, they are assumed to remain in a fast equilibrium. The state of a $TTWWAA$ molecule can be determined by the 3

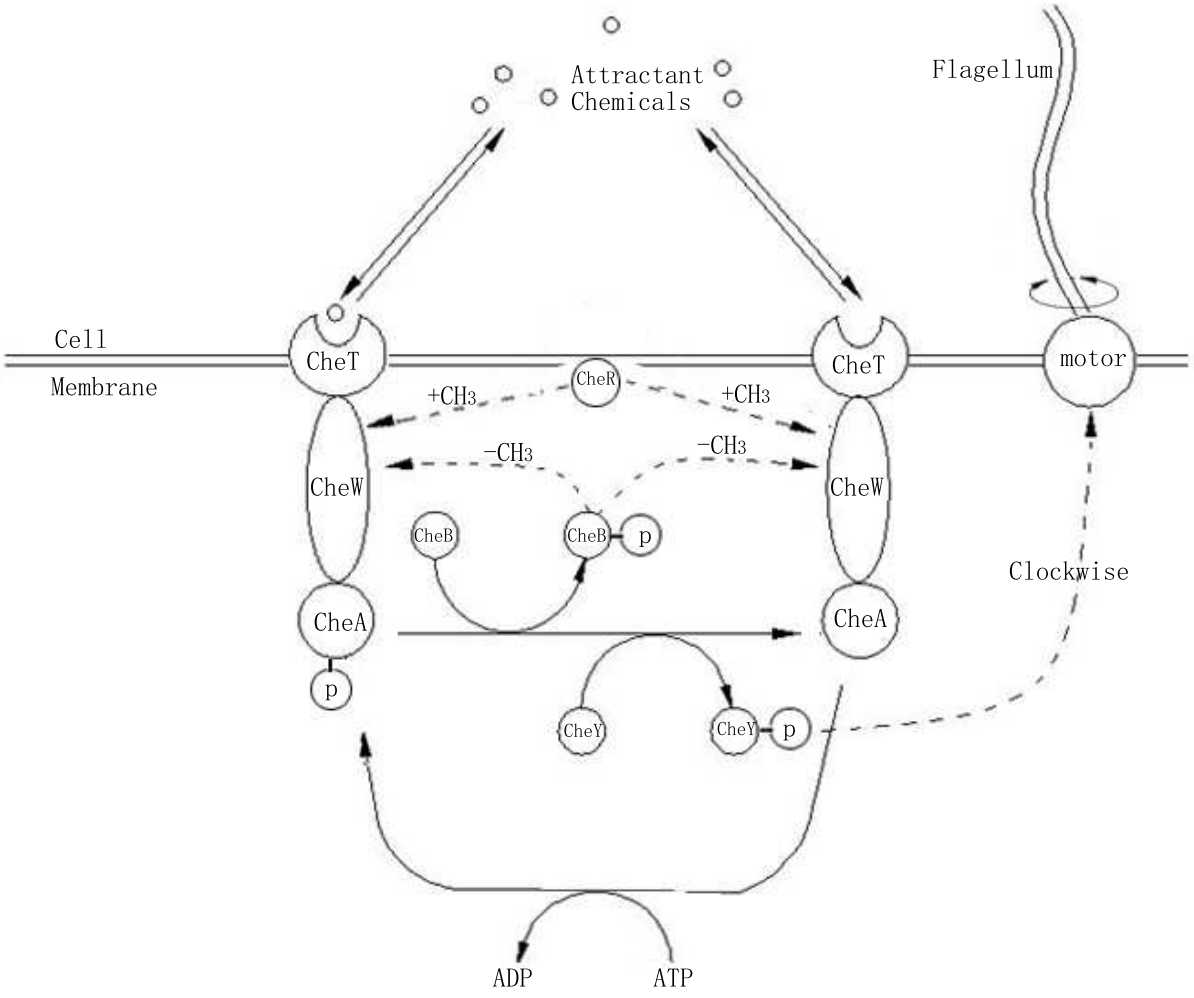

Figure 5.2: Chemical reacting network of bacterial chemotaxis.

properties above and represented by combinations of the capital letters and numbers in brackets. For example, a state "$B_p2?$" indicates that the molecule is bound to *CheBp*, with a methylation level 2, and either active or inactive. The reaction list of this model without equilibria is given in Table 5.1.

Description	*Reaction*	*Reactant TTWWAA State*	*Product TTWWAA State*
CheBp (un)binding	$TTWWAA + B_p \leftrightarrow TTWWAA$	$N?A$	$B_p?A$
De-methylation	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p1?$	$N0?$
	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p2?$	$N1?$
	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p3?$	$N2?$
	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p4?$	$N3?$
CheR (un)binding	$TTWWAA + R \leftrightarrow TTWWAA$	$N?I$	$R?I$
Methylation	$TTWWAA \rightarrow TTWWAA + R$	$R0?$	$N1?$
	$TTWWAA \rightarrow TTWWAA + R$	$R1?$	$N2?$
	$TTWWAA \rightarrow TTWWAA + R$	$R2?$	$N3?$
	$TTWWAA \rightarrow TTWWAA + R$	$R3?$	$N4?$

Table 5.1: Reaction list of the simplified bacterial chemotaxis chemical model.

We also translated it into the corresponding representation for the SSA. While keeping the fast equilibria, we treat each state of $TTWWAA$ as one variable and the number of reactions increases from 12 to 28. Table 5.2 provides a list of all reactions in the converted model. In the list, the subscript of a $TTWWAA$ refers to the methylation level.

We implemented StochSim and the SSA with the respective models above. The experimental results are shown in Section 2.5.1. To make the simulation more complete and realistic, we completed the simplified model above by including all the reactions shown in Figure 5.2. In this model, 5 binding sites are considered on $TTWWAA$ as explained in the following.

1. Bound to *CheR* (R), bound to *CheBp* (B_p), or not bound (N).
2. Level of methylation (0, 1, 2, 3, 4).
3. Active (A) or inactive (I).

Description	*Reaction*
CheBp (un)binding	$TTWWAA_0 + B_p \leftrightarrow TTWWAA_0B_p$
	$B_p + TTWWAA_1 \leftrightarrow TTWWAA_1B_p$
	$B_p + TTWWAA_2 \leftrightarrow TTWWAA_2B_p$
	$B_p + TTWWAA_3 \leftrightarrow TTWWAA_3B_p$
	$B_p + TTWWAA_4 \leftrightarrow TTWWAA_4B_p$
De-methylation	$TTWWAA_1B_p \rightarrow TTWWAA_0 + B_p$
	$TTWWAA_2B_p \rightarrow TTWWAA_1 + B_p$
	$TTWWAA_3B_p \rightarrow TTWWAA_2 + B_p$
	$TTWWAA_4B_p \rightarrow TTWWAA_3 + B_p$
CheR (un)binding	$R + TTWWAA_0 \leftrightarrow TTWWAA_0R$
	$R + TTWWAA_1 \leftrightarrow TTWWAA_1R$
	$R + TTWWAA_2 \leftrightarrow TTWWAA_2R$
	$R + TTWWAA_3 \leftrightarrow TTWWAA_3R$
	$R + TTWWAA_4 \leftrightarrow TTWWAA_4R$
Methylation	$TTWWAA_0R \rightarrow TTWWAA_1 + R$
	$TTWWAA_1R \rightarrow TTWWAA_2 + R$
	$TTWWAA_2R \rightarrow TTWWAA_3 + R$
	$TTWWAA_3R \rightarrow TTWWAA_4 + R$

Table 5.2: Translated reaction list of the simplified bacterial chemotaxis model for the SSA.

4. Phosphorylated (P) or not (N).

5. Bound to *CheY* (Y), bound to *CheB* (B), or not bound (N).

In Table 5.3 we list all the reactions of the complete model in the same way of representation as in Table 5.1.

Description	*Reaction*	*Reactant TTWWAA State*	*Product TTWWAA State*
Auto-phosphorylation	$TTWWAA \rightarrow TTWWAA$	$??AN?$	$??AP?$
De-phosphorylation	$B_p \rightarrow B$	-	-
CheB unbinding	$TTWWAA \rightarrow TTWWAA + B$	$???NB$	$???NN$
CheB binding	$TTWWAA + B \rightarrow TTWWAA$	$????N$	$????B$
CheBp (un)binding	$TTWWAA \leftrightarrow TTWWAA + B_p$	$???PB$	$???NN$
CheBp binding	$TTWWAA + B_p \rightarrow TTWWAA$	$N1A??$	$B_p1A??$
	$TTWWAA + B_p \rightarrow TTWWAA$	$N2A??$	$B_p2A??$
	$TTWWAA + B_p \rightarrow TTWWAA$	$N3A??$	$B_p3A??$
	$TTWWAA + B_p \rightarrow TTWWAA$	$N4A??$	$B_p4A??$
CheBp unbinding	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p????$	$N????$
De-methylation	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p1???$	$N0???$
	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p2???$	$N1???$
	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p3???$	$N2???$
	$TTWWAA \rightarrow TTWWAA + B_p$	$B_p4???$	$N3???$
CheR binding	$TTWWAA + R \rightarrow TTWWAA$	$N0I??$	$R0I??$
	$TTWWAA + R \rightarrow TTWWAA$	$N1I??$	$R1I??$
	$TTWWAA + R \rightarrow TTWWAA$	$N2I??$	$R2I??$
	$TTWWAA + R \rightarrow TTWWAA$	$N3I??$	$R3I??$
CheR unbinding	$TTWWAA \rightarrow TTWWAA + R$	$R????$	$N????$
Methylation	$TTWWAA \rightarrow TTWWAA + R$	$R0???$	$N1???$
	$TTWWAA \rightarrow TTWWAA + R$	$R1???$	$N2???$
	$TTWWAA \rightarrow TTWWAA + R$	$R2???$	$N3???$
	$TTWWAA \rightarrow TTWWAA + R$	$R3???$	$N4???$
(De-)phosphorylation	$Y_p \leftrightarrow Y$	-	-
CheY unbinding	$TTWWAA \rightarrow TTWWAA + Y$	$???NY$	$???NN$
CheY binding	$TTWWAA + Y \rightarrow TTWWAA$	$????N$	$????Y$
CheYp (un)binding	$TTWWAA \leftrightarrow TTWWAA + Y_p$	$???PY$	$???NN$

Table 5.3: Reaction list of the complete bacterial chemotaxis chemical model.

5.3 Numerical Experiments on the Complete Model

With the complete model in Table 5.3, the choice of the simulation method is still an issue. Since the total number of reactions increases to 30, efficiency becomes a big concern. First of all, since this model is completed based on the simplified model simulated by StochSim, StochSim is always an option. However, as we can easily calculate, the total number of states of $TTWWAA$ is $3 \times 5 \times 2 \times 2 \times 3 = 180$, which means there would be 180 state variables derived from the same species in the model for the SSA. Then the number of reactions in the SSA model would be over 500. The SSA apparently does not qualify as a good candidate for simulation.

It is not difficult to notice that the reactions in Table 5.3 all follow the exact form of the example (4.1). They are perfect to be considered as rules and the model is actually a rule-based model. Therefore, all the methods in the previous chapter turn to be available for the model now. Note that, since the rule-based simulation methods do not handle any rule with multiple rates, the first rule in the table has to be expanded to 5 rules as in Table 5.4, making a total of 34 rules. This is because the equilibrium of the activation and deactivation is affected by the methylation level.

Description	*Reaction*	*Reactant TTWWAA State*	*Product TTWWAA State*
Auto-phosphorylation	$TTWWAA \rightarrow TTWWAA$	?0AN?	?0AP?
	$TTWWAA \rightarrow TTWWAA$	?1AN?	?1AP?
	$TTWWAA \rightarrow TTWWAA$	?2AN?	?2AP?
	$TTWWAA \rightarrow TTWWAA$	?3AN?	?3AP?
	$TTWWAA \rightarrow TTWWAA$	?4AN?	?4AP?

Table 5.4: The 5 replacement rules for the first rule in the complete bacterial chemotaxis chemical model.

Experiments were first conducted on the complete model with the original StochSim package. Then we implemented the PNFA and FSSSA on this model. The CPU times for different method are listed in Table 5.5.

	Intel Pentium4 2.80GHz	*iMac Core 2 Duo 2.0GHz*	*Average Step Size*
StochSim	*50.53*	*28.63*	1.97×10^{-7}
PNFA	*21.35*	*16.63*	7.51×10^{-6}
FSSSA	*9.32*	*8.01*	7.53×10^{-6}

Table 5.5: Comparison of CPU times in seconds on the complete bacterial chemotaxis model for 1 run, on 2 different machines.

The simulation results can be explained with the time complexity analysis in Table 4.1. The first reason why StochSim exhibits the lowest efficiency is its step size, which is about 38 times smaller than those of the other two methods. Moreover, we used the original StochSim package, which definitely adds certain overheads. On the other hand, because the PNFA and FSSSA share the same rule list, they have similar step sizes. The difference is the cost in each step. According to our analysis in Section 4.4, it is reasonable that the FSSSA has better performance. Since in this model N_{obj} is over $4,000$, the NFA executes much more operations in maintaining the reactant pointer lists.

5.4 Graphical Interface

In order to deliver more realistic simulations of the bacterial chemotaxis, we developed a graphical interface in C++ to visualize the movements of multiple bacteria. This interface has a two-dimensional environment with 100×100 grids. The user can specify the environment by assigning different chemical concentrations to different areas of grids. The chemicals are categorized into food (attractant) chemical and poison (repellent) chemical. Diffusion of both can be turned on and off during simulation. Inside the environment, multiple bacteria are simulated with their movements through grids. Figure 5.3 shows a screen print of the interface.

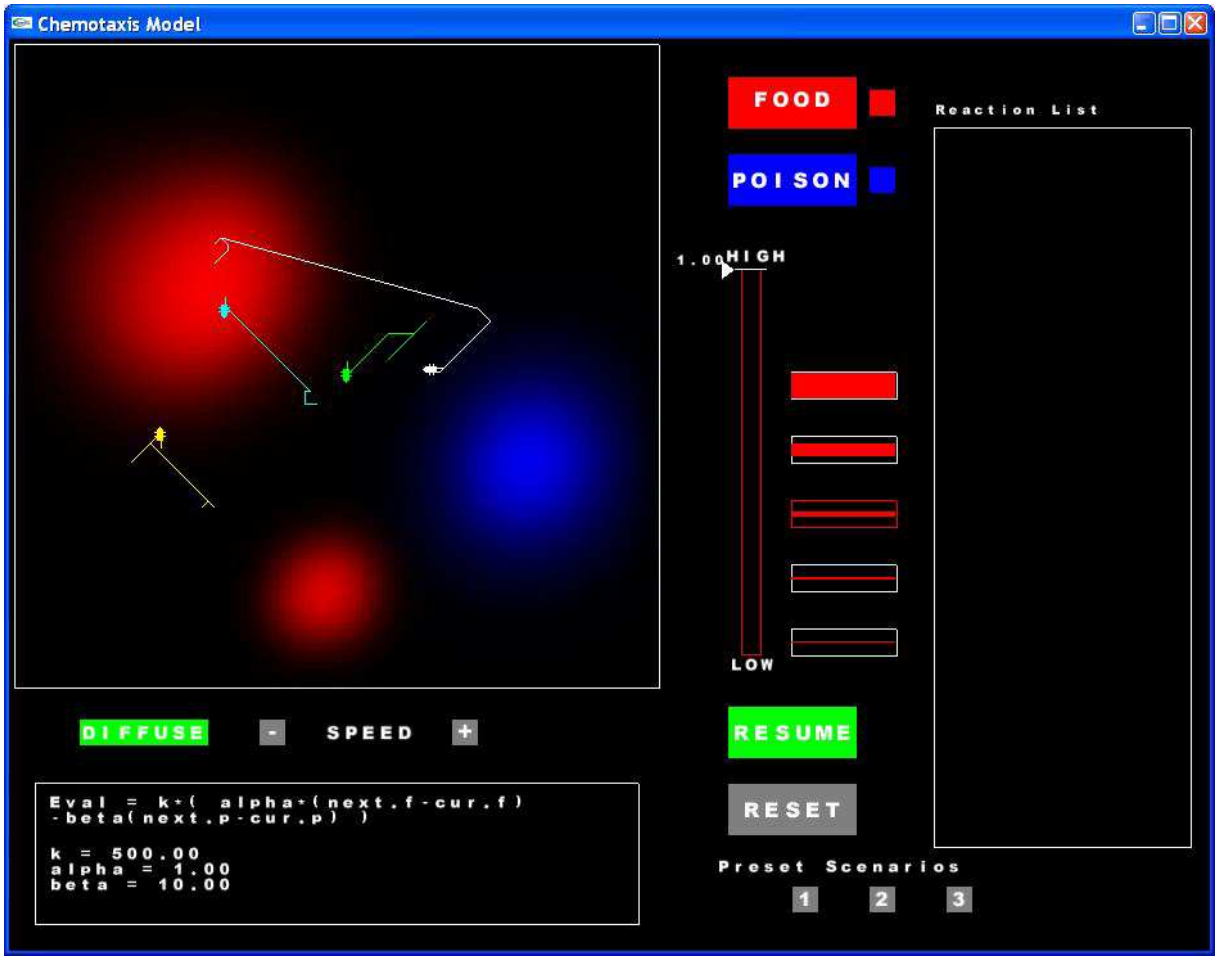

Figure 5.3: Screen print of the graphical interface.

5.4.1 Models of Mechanisms

For the purpose of comparison, 4 bacteria are placed in the environment and can direct themselves using different mechanisms. The mechanisms can decide how a bacterium selects its moving direction for the next step. They include:

1. Random walk model.
2. Run/tumble model.
3. Chemical model I in Table 5.2.
4. Chemical model II in Table 5.3.

The random walk model is a mathematical model with the assumption that the bacterium directs itself with simple possibilities. In each simulation step, the bacterium first compares the local chemical concentrations with the previous ones before its last movement by calculating $Eval$ given by

$$Eval = k(\alpha(F_{cur} - F_{pre}) - \beta(P_{cur} - P_{pre})) \tag{5.1}$$

where k, α and β are parameters, F_{cur} and P_{cur} are respectively the local chemical concentrations of food and poison, and F_{pre} and P_{pre} are the concentrations at the previous area before the last movement. $Eval$ is used to indicate the potential that the bacterium should remain its original direction for its next movement. Then, 8 possibilities are assigned to the 8 possible directions as follows. The possibility that the bacterium will keep its original direction is calculated as in (5.2).

$$p_{original} = \{ \begin{array}{c} if\ 1 + Eval \geq 0,\ \frac{1+Eval}{8+Eval} \\ otherwise,\ 0 \end{array} \}. \tag{5.2}$$

The possibility that the bacterium will move in each of the other directions is given in (5.3).

$$p_{other} = \{ \begin{array}{c} if\ 1 + Eval \geq 0,\ \frac{1}{8+Eval} \\ otherwise,\ \frac{1}{7} \end{array} \}. \tag{5.3}$$

Last, a random number is generated and decides the direction for the next movement.

Comparing to the first model, the run/tumble model additionally incorporates the swimming pattern of the bacterium. It is based on experimental observations that the bacterium tends to run in a straight line when it arrives at a better environment and tumbles at a worse environment [13]. To simulate this kind of behaviors, we suppose the bacterium is in either the run mode or the tumble mode, and consider the transitions between the run mode and tumble mode as one reversible reaction

$$tumble \underset{k_r}{\overset{1.0}{\rightleftharpoons}} run, \tag{5.4}$$

where the reaction rate of the forward reaction is a constant and the reverse rate k_r is calculated by $\frac{0.1}{1+Eval}$. In each simulation step, the model first calculate $Eval$ as in the first model, and then simulates the reaction (5.4) stochastically. If the final mode is run, the bacterium keeps its original direction. If the final mode is tumble, a new direction except the original one is randomly selected.

The third model replaces the simple modeling equations (5.1), (5.2) and (5.3) with the simplified chemical network of bacterial chemotaxis, which was first simulated by StochSim. For the consideration of efficiency, this chemical model is translated as in Table 5.2 for the SSA. In each simulation step, the chemical network is simulated with the SSA and then outputs the population of the active $TTWWAA$. Then, instead of simulating reactions in (5.4), a response curve is used to determine the possibility that the bacterium will tumble. An example of a response curve can be found in Figure 5.4. Last, a random number is compared to the possibility and decides whether the bacterium will tumble or keep moving in its original direction.

The last model uses the full chemical network as in Table 5.3. According to the comparison in Table 5.5, this chemical network is simulated by the FSSSA. The simulation is the same as that of the third model except that the output becomes the population of $CheYp$ and another response curve is used.

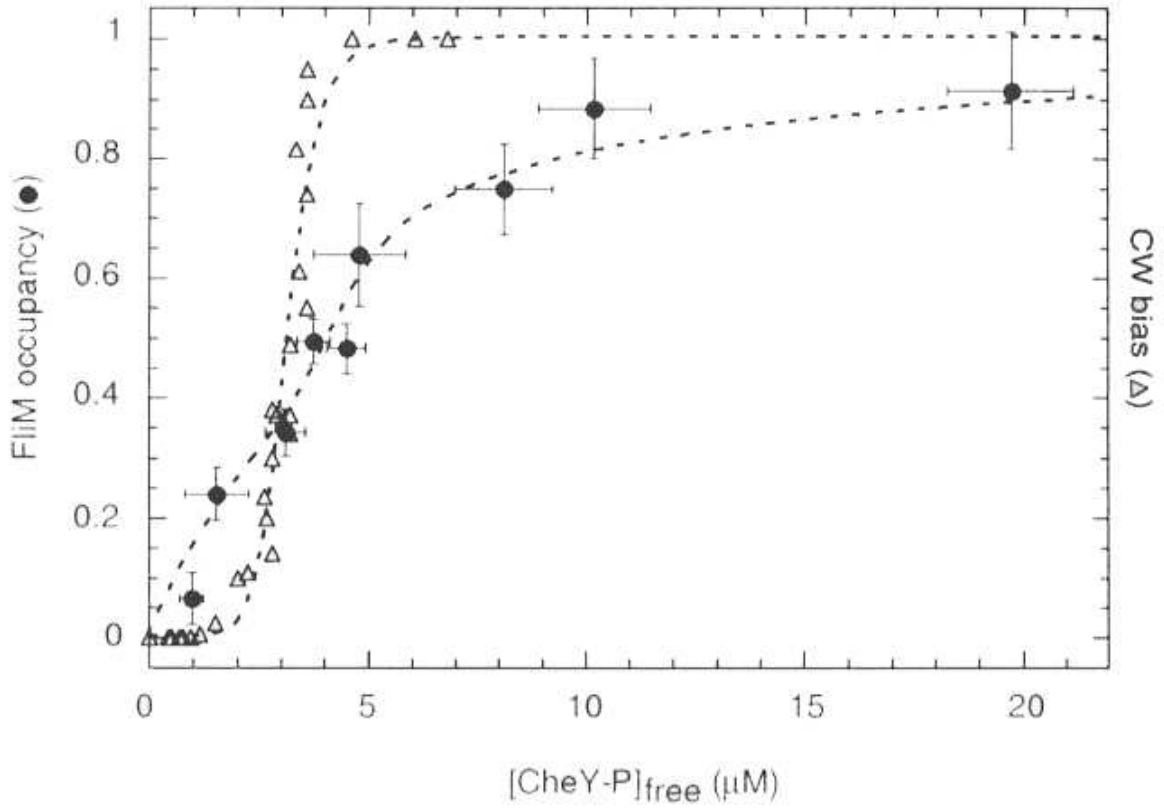

Figure 5.4: The response curve taken from [13]. The x-axis is concentration of *CheYp*. The y-axis can be interpreted as the corresponding possibility that the bacterium tumbles.

5.4.2 Experiments

We compared the 4 models by analyzing the behavior of the four bacteria in a sample environment. This environment is specified with only food chemical concentrations and does not allow chemical diffusion. The concentrations gradually increases from 0.0 at the bottom of the environment to the maximum value at the top. During simulation, 4 bacteria are placed in the same grid near the bottom. Then we count the average numbers of steps that different bacteria take to arrive at the top area. See results in Table 5.6.

Model	*Average Step Number*
Random Walk	*479*
Run/Tumble	*106*
Chemical I	*99*
Chemical II	*128*

Table 5.6: Numbers of steps that the four model bacteria take to arrive at the top area. The results are the average numbers for 1,000 simulations.

The results show that, although the mathematical mechanism for the random walk model is simple and directly responds to local concentrations, it does not represent the most efficient way to find the more highly concentrated area. However, under the second model with the more realistic run/tumble pattern, the simulated bacteria exhibit obvious higher ability in locating food. The last two chemical models have similar performance with the second one. As we can see, the run/tumble moving strategy has the advantage by keeping the original direction while the bacterium moves towards the higher concentrations.

Chapter 6

Conclusion and Future Work

In this thesis we analyze the theoretical foundation of StochSim and compare the SSA and StochSim. We have demonstrated that when Δt is very small, StochSim can be viewed as a first-order approximation of the SSA. In general the SSA is more efficient than StochSim, especially when dealing with systems with a large N (total population). StochSim has advantages when multi-state species are involved in the system. We have proposed the HSSA method to improve the efficiency of the SSA in the multi-state case.

Based on the comparison between StochSim and SSA [23], we have proposed a hybrid method that possesses the advantages of both StochSim and SSA. We first improved StochSim by taking adaptive step sizes and eliminating pseudo-molecules. Then by partitioning the system into two subsystems, the SSA and the improved StochSim are combined. This hybrid method is tested with numerical examples and shows great performance improvement. A more complicated model is under construction for the numerical test of our hybrid method.

As another method of particle-based modeling, the rule-based models have shown advantages in representing systems involving multi-state situations [26]. The NFA is a particle-based method for stochastic simulation of the rule-based models. In this thesis, we have proposed the PNFA as a hybrid method combining the population-based and particle-based schemes,

and the FSSSA which is a population-based method. Both methods can be used for the rule-based model simulation.

At last, we have used the bacterial chemotaxis as a biological example. We have successfully implemented the PNFA and FSSSA on the complete chemical rule-based model of bacterial chemotaxis. We have also constructed a graphical interface to visualize the simulation. Our experiments have demonstrated that bacteria with the run and tumble pattern have better ability in locating more highly concentrated food areas.

Future research should be focused on the development of suitable simulation methods for the rule-based models. We are trying to answer questions such as, how should different kinds of chemical networks be modeled? How should we choose from the particle-based, population-based and the hybrid schemes for different models? To begin with, we will first conduct some more detailed analysis on the nature of the NFA, PNFA and FSSSA. We believe our research presented here can help the exploration for a best simulation method for rule-based models.

Bibliography

[1] H. McAdams, A. Arkin. Stochastic mechanisms in gene expression. *Proc. Natl. Acad. Sci. USA*, 94:814–819, 1997.

[2] A. Arkin, J. Ross and H. McAdams. Stochasitc kinetic analysis of developmental pathway bifurcation in phage λ-infected Escherichia coli cells. *Genetics*, 149:1633–1648, 1998.

[3] N. Fedoroff, W. Fontana. Small number of big molecules. *Science*, 297:1129–1131, 2002.

[4] M. Elowitz, A. Levine, E. Siggia, P. Swain. Stochastic gene expression in a single cell. *Science*, 297:1183–1186, 2002.

[5] D. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.*, 22:403–434, 1976.

[6] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–2361, 1977.

[7] M. Gibson, J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104:1876, 2000.

[8] Y. Cao, H. Li, L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J. Chem. Phys.*, 121:4059–4067, 2004.

[9] J. McCollum, G. Peteron, C. Cox, M. Simpson, N. Samatova. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Comput. Biol. Chem.*, 30:39–49, 2006.

[10] D. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115:1716, 2001.

[11] W. Hlavacek, J. Faeder, M. Blinov, A. Perelson, B. Goldstein. The complexity of complexes in signal transduction. *Biotechnol. Bioeng.*, 84:783–794, 2003.

[12] D. Bray. Genomics: Enhanced: Molecular prodigality. *Science*, 299:1189–1190, 2003.

[13] H. Berg. *E. coli in motion*. New York: Springer, 2004.

[14] T. Schulze. Kinetic Monte Carlo simulations with minimal searching. *Phys. Rev. E*, 65:036704, 2002.

[15] H. Li, L. Petzold. Logarithmic direct method for discrete stochastic simulation of chemically reacting systems. *Technical Report*, 2006.

[16] A. Chatterjee, D. Vlachos. An overview of spatial microscopic and accelerated kinetic Monte Carlo methods. *J. Comput. Aided Mater. Des.*, 14:253–308, 2007.

[17] A. Regev, E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419, 2002.

[18] A. Regev, E. Panina, W. Silverman, L. Cardelli, E. Shapiro. BioAmbients: An abstraction for biological compartments. *Theoretical Computer Science*, 325:141–167, 2004.

[19] C. Morton-Firth, D. Bray. Predicting temporal fluctuations in an intracellular signalling pathway. *J. theor. Biol.*, 192:117–128, 1998.

[20] C. Morton-Firth, T. Shimizu, D. Bray. A free-energy-based stochastic simulation of the rat receptor complex. *J. Mole. Bio.*, 287:1059–1074, 1999.

[21] T. Shimizu, D. Bray. *Computational Cell Biology - the Stochastic Approach in Foundations of Systems Biology*. MIT Press, Cambidge, MA, 2001.

[22] N. Novere, T. Shimizu. StochSim: Modeling of stochastic biomolecular processes. *Bioinfomatics*, 17:575–576, 2001.

[23] Z. Liu, Y. Cao. Detailed comparison between the StochSim and SSA. *IET Systems Biology*, 2:334–341, 2008.

[24] M. Pettigrew, H. Resat. Modeling signal transduction networks: A comparison of two stochastic kinetic simulation algorithms. *J. Chem. Phys.*, 123:114707, 2005.

[25] W. Hlavacek, J. Faeder, M. Blinov, R. Posner, M. Hucka, W. Fontana. Rules for modeling signal-transduction systems. *Sci. STKE*, page re6, 2006.

[26] J. Faeder, M. Blinov, W. Hlavacek. *Rule-based modeling of biochemical systems with BioNetGen*. In methods in molecular biology: Systems biology, Humana Press, Totowa, NJ, i. v. maly edition, 2008.

[27] V. Danos, J. Feret, W. Fontana, J. Krivine. Scalable simulation of cellular signaling networks. *Lect. Notes Comput. Sci.*, 4807:139–157, 2007.

[28] J. Yang, M. Monine, J. Faeder, W. Hlavacek. Kinetic monte carlo method for rule-based modeling of biochemical networks. *Physical Review*, 78:031910, 2008.

[29] M. Sneddon, W. Pontius, J. Faeder, T. Emonet. NFsim: Managing complexity in stochastic simulation of reaction networks. The 2nd q-bio conference, Santa Fe, NM, 2008.

[30] D. Bray's Group. http://www.pdn.cam.ac.uk/groups/comp-cell/StochSim.html.

[31] L. Lok, R. Brent. Automatic generation of cellular reaction networks with Moleculizer 1.0. *Nat. Biotechnol.*, 23:131–136, 2005.

[32] iGEM Team Heidelberg. http://2008.igem.org/Team:Heidelberg/Human-Practice/Phips-the-Phage/General-Backround.